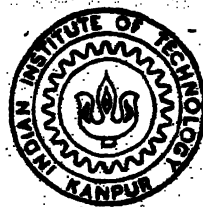


ROBOT BEHAVIOUR CONFLICTS: CAN INTELLIGENCE BE MODULARIZED?

by

MALI AMOL DATTATRAYA



99u/M

2-r

**DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR**

April, 1994



ROBOT BEHAVIOUR CONFLICTS: CAN INTELLIGENCE BE MODULARIZED?

*A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY*

by
MALI AMOL DATTATRAYA

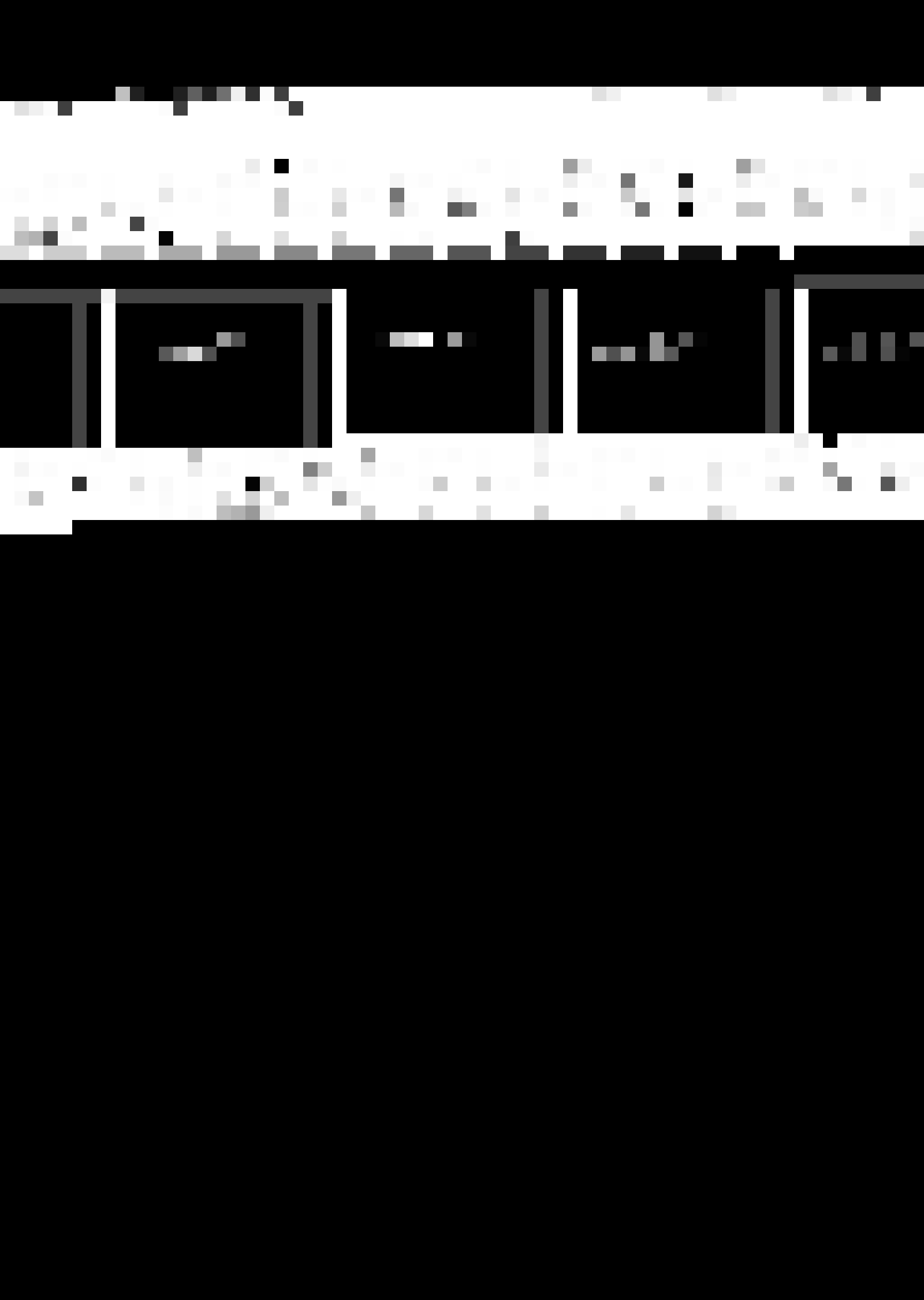
to the
**DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR
April 1994**



ME-1994-M-DAT-ROB

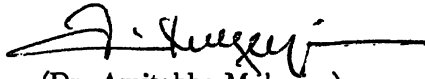
30 MAY 1994 / ME
GENERAL LIBRARY
I I T, KANPUR
Acc. No. A. 117809

TH
629.892
D262A



CERTIFICATE

This is to certify that the work contained in the thesis titled **ROBOT BEHAVIOUR CONFLICTS: CAN INTELLIGENCE BE MODULARIZED?** has been carried out by Mali Amol Dattatraya under my supervision and it has not been submitted elsewhere for a degree.



(Dr. Amitabha Mukerjee)

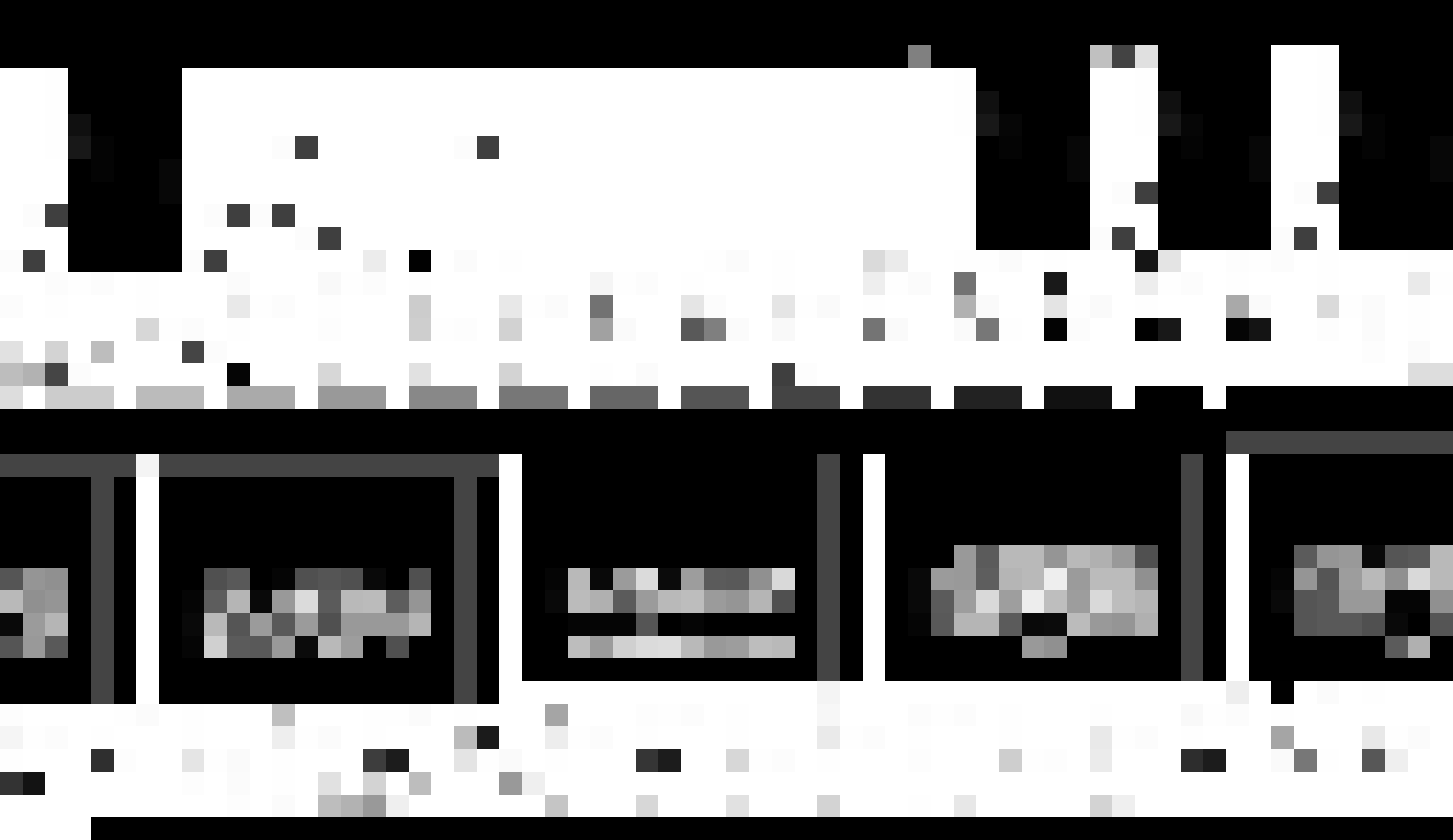
Assistant Professor,

Dept. of Mechanical Engg., IIT Kanpur.



ABSTRACT

In this paper, we examine one of the fundamental assumptions of behaviour-based models: that complex functionalities can be achieved by decomposition into simpler behaviours. In particular we look at the issue of conflicts among robot behaviour modules. The chief contribution of this work is a formal characterization of temporal cycles in behaviour systems and the development of an algorithm for detecting and avoiding such conflicts. We develop the mechanisms of stimulus specialization and response generalization for eliminating conflicts. Thus the probable conflicts can be detected and eliminated before implementation. However the process of cycle elimination weakens the behaviour structure. We show how (a) removing conflicts results in less flexible and less useful behaviour modules and (b) the probability of conflict is greater for more powerful behaviour systems. We investigate several other behaviour representations and show that these are also susceptible to cyclic conflicts since conflicts occurring at knowledge level cannot be solved at the representation level. We conclude that purely reactive systems are limited by cyclic behaviours in the complexity of tasks they can perform. Possible solutions involve hybrid systems with internal state that can mediate conflicts, and possibly, a capacity for learning or self-modification.



ACKNOWLEDGEMENT

I am greatly indebted to Dr. Amitabha Mukerjee for the deep interest in my work that he took during the supervision of this thesis. I thank him for enthusiastically replying hundreds of my mails and also editing my work a large number of times with a great care and interest. His contribution to this work has been much more than a supervision. Indeed I count myself very lucky to get a thesis supervisor like him.

I thank Dr. Harish Karnick, Dr. S. Mukherjee, Dr. N. K. Sharma, Dr. Lilavati Krishnan, Dr. Bijoy Boruah and Dr. George Kurian for providing helpful comments on the earlier drafts of my papers.

I thank Soumya Bhattacharya, Anirvan Dasgupta and Samiran Mandal for their advice and help. Thanks to all who have directly or indirectly helped me in making my stay at IIT Kanpur enjoyable.

Mali Amol Dattatraya.

18 April 1994.



Dedicated To
Dr. Amitabha Mukerjee

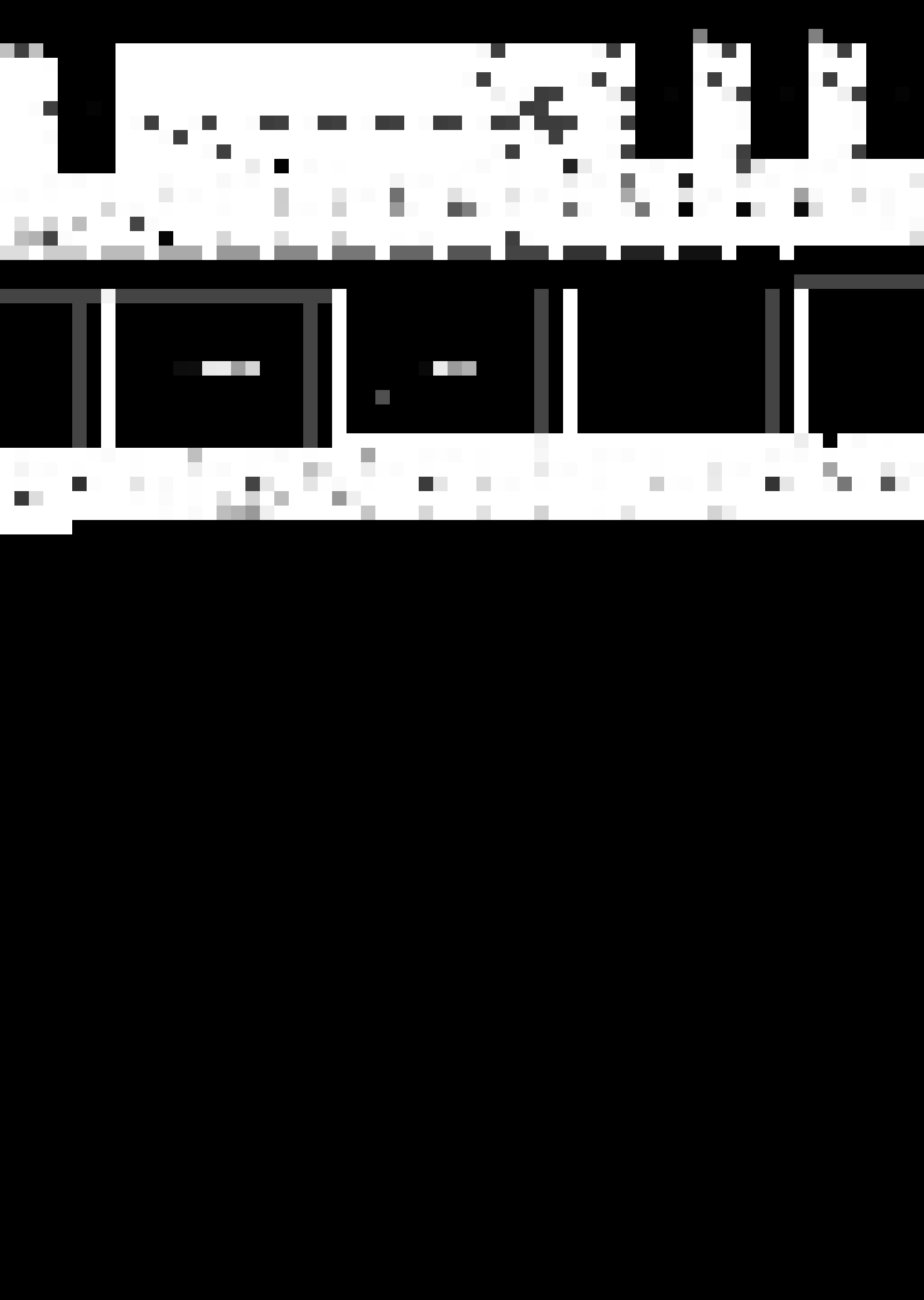


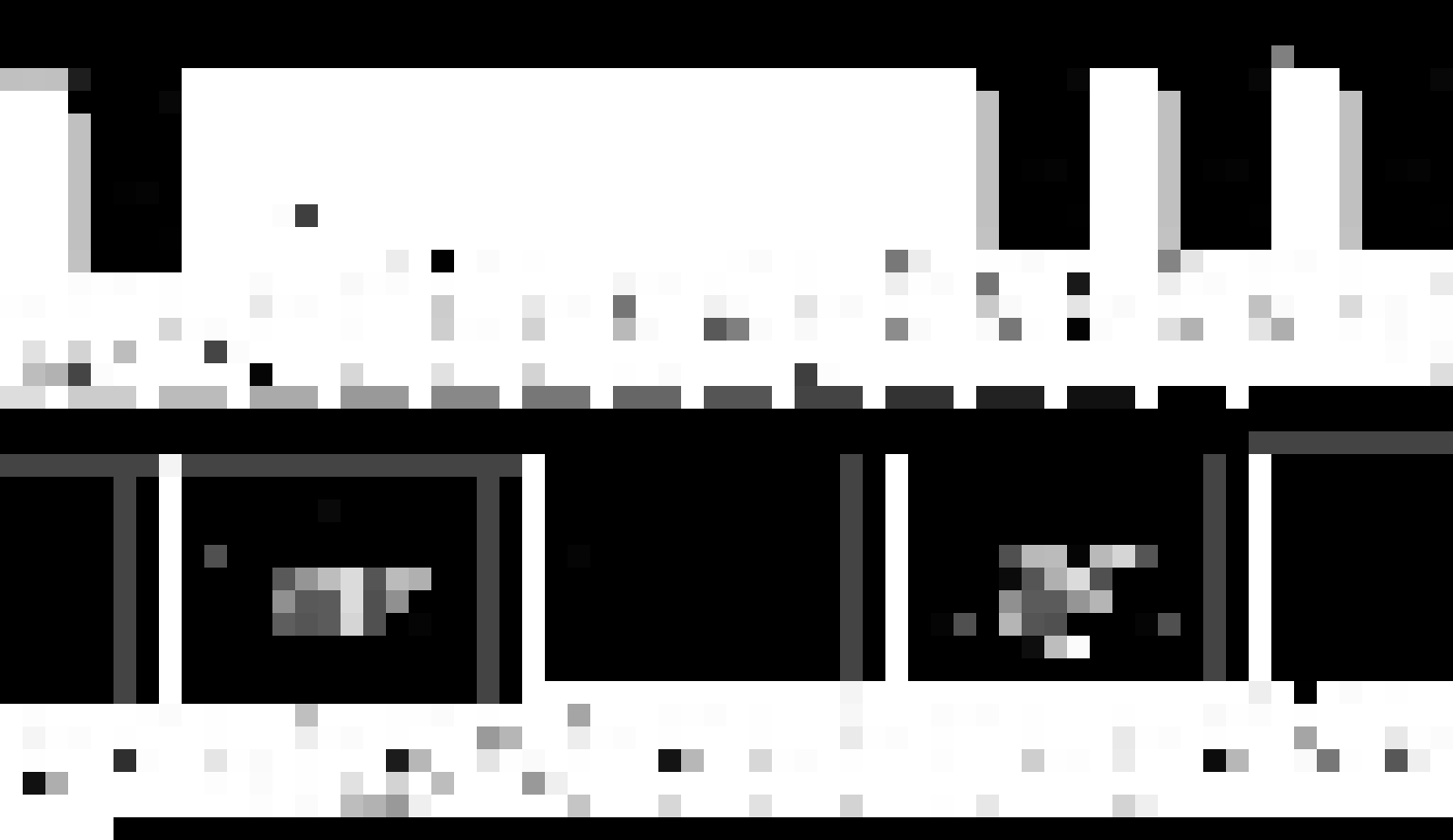
TABLE OF CONTENTS

Chapter 1	Introduction	1
Chapter 2	What Is a Behaviour?	6
	2.1 Notation	7
	2.2 Behaviour Chain	8
	2.3 Frame Problem	10
	2.4 Power, Usefulness and Flexibility of Behaviours	11
chapter 3	Detection of Conflicts.	13
	3.1 Terminated Cycles	14
	3.2 Prioritization	15
	3.3 Cyclic Conflicts and Behaviour Graphs	18
	3.4 Conflict Detection Algorithm	19
Chapter 4	Behaviour Refinement	21
	4.1 Stimulus Specialization.	23
	4.2 Response Generalization	24
	4.3 Effects of Behaviour Refinement	25
	4.4 Residual	29
Chapter 5	Similarities with AI Planning.	31
Chapter 6	Alternate Architectures	33
	6.1 Potential Fields	33
	6.2 Fuzzy Logic	34
	6.3 Connectionist Architectures	35
	6.4 Hybrid Systems	36
	6.5 Meta Rules	36
	6.6 Learning and Behaviour Modification.	37
Chapter 7	Conclusion	39
	References	41



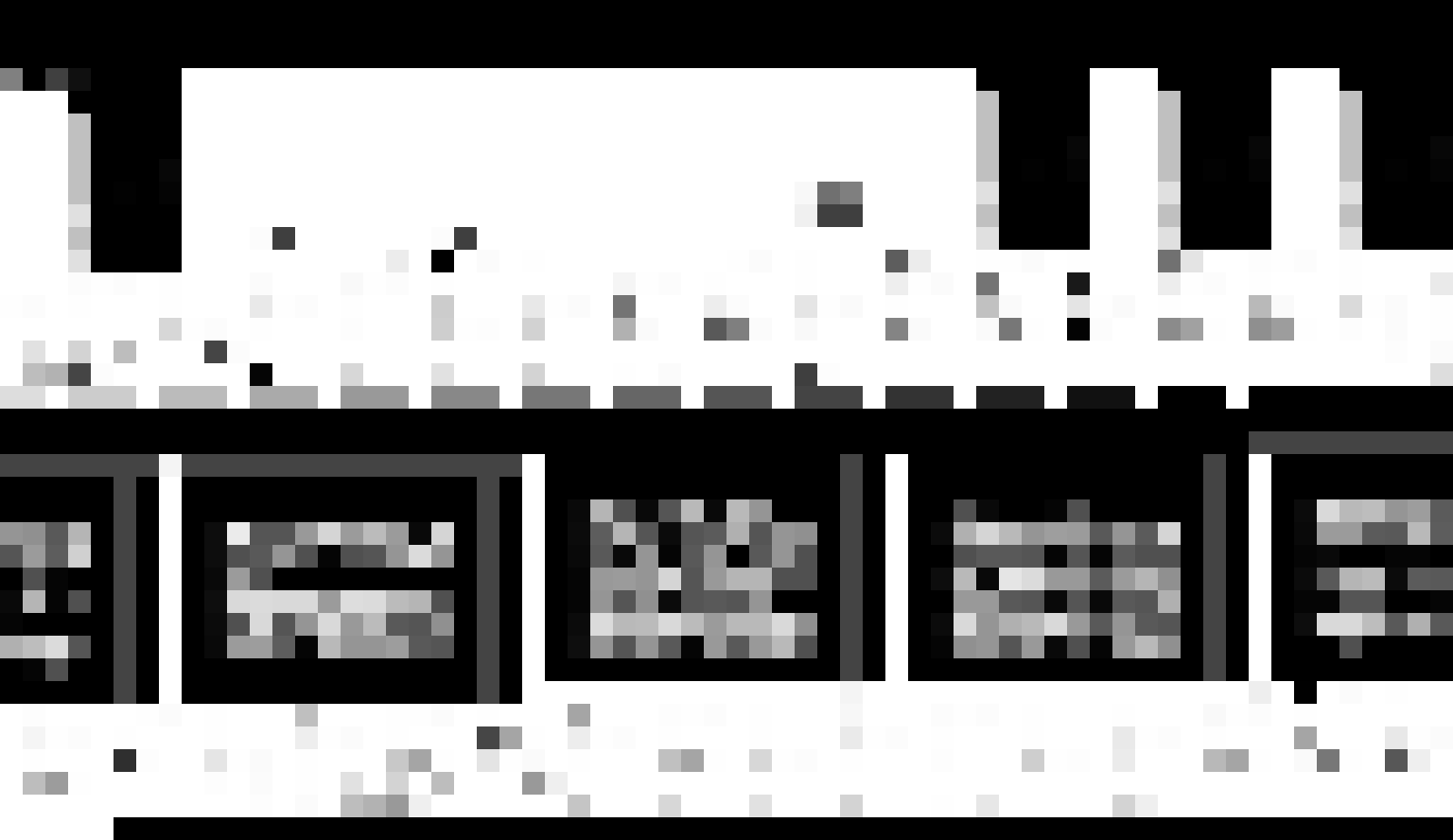
LIST OF FIGURES

Figure No.	Figure Description.	Page No.
1	Conflict in picking and placing the can.	4
2	Behaviour chain with concurrent behaviours.	9
3	Cyclic conflict in a temporal chain of behaviours.	14
4	Termination condition in a recursive cycle.	15
5	Conflict that can be prevented by retriggerable monostable.	18
6	Graphical representation of the modules and the cycles in it.	20



into a problem solving system; whatever systems have been implemented have been done by human programmers. In particular, one must look at the mechanisms for combining behaviours, what Brooks calls “*controlled amalgamation*” [19]. Also the lack of a representation prevents the system from doing tasks that require knowledge that cannot be obtained through perception - such as analogous examples, or other tasks involving creativity. Mataric has observed that subsumption architecture and fully reactive systems have been limited to applications requiring no explicit internal representation, which imposed a fundamental limitation on the domain of applications for the architecture [40]. Hartley & Pipitone seem to think that the subsumption architecture as currently defined is not sufficiently modular and it is not always clear if a strict hierarchy of behaviours is appropriate [27]. Payton [46] argues that information is invariably lost when it is hidden within behaviour modules or contained in commands which are subsumed. Later in this paper, in *Chapter 5*, we discuss the relation between reactive behaviours and planning - whether, as some suspect, the evils of planning (cycles, interdependent goals) may also arise in reactive architectures [24].

One of the problems with modular designs (databases, architectures, factories) is that of conflicting functionalities between modules. Such inter-modular conflict often arises due to resource-sharing type of problems, which can often be resolved by temporally sequencing the behaviours in a certain manner. This is achieved in traditional behaviour systems by *suppressing* one behaviour by another. In most implementations, control of behaviours has been done in an *ad hoc* manner, with suppressive links being added whenever it suited the primary objective of demonstrating success in the current task objectives. Brooks [13] stresses that additional layers can be added later, and the initial working system need never be changed - our results show that such a claim is most probably not tenable. The number of behaviours possible from a given set of behaviours combina-



torially explodes if spatial and temporal ordering is allowed [1]. Then how does one order or search this space? The more basic question raised by this is: *How does one put the behaviour modules together and get useful performance?* A related issue is extending or editing behaviour systems: the question that this raises is the issue of compatibility between the new and the pre-existing behaviour modules. All these questions depend on identifying the possible sources of inter-modular conflicts that are likely to arise in behaviour chains.

This paper is one of the first formal investigations on the issue of combining behaviours and inter-behaviour conflicts. Despite the large scale attention such models have been receiving, the issue of interbehaviour conflict, which challenges the fundamental assumption of behaviour modularity, has not received sufficient attention. A typical scenario is one where the consequence of a behaviour A triggers the stimulus for behaviour B which precedes A, resulting in an unending cycle. Connell [18] too has reported that without a history of events or an overall picture of the situation, the robot faces the possibility of getting stuck in local minima or entering infinite loops instead of achieving its true goal. Such conflicts are also beginning to show up in the more complex behaviour implementations. For example, Connell [18] records an instance where a can collecting robot attempts to re-pick the can it has just deposited in the destination area as shown (Figure 1). This conflict was not detected until after a full implementation. Cyclic conflict of going back and forth between two obstacles has been reported [42]. Cyclical wandering has been reported by Anderson & Donath [1]. Cycle has been reported in the task of co-operating robots pushing a box [33]. Can behaviour systems be constructed so that such conflicts can be detected beforehand? Can one construct an algorithm for modifying behaviours so as to avoid such conflicts? These are some of the questions we set out to answer.

Behaviour systems have also been moving away from the purely reactive paradigm. A well-



known extension of the Brooks' approach includes the SONAR MAP [9] which is a module that learns what looks suspiciously like a central representation. Chatila [17] points out that sooner or later in the evolution of this system, the ability to *plan* actions and execute them (and not only to react to external stimuli) will have to be incorporated to achieve increased autonomy. Connell [18] concludes that relaxing the temporal locality condition, the robot's navigational skills could be significantly augmented with the addition of a relatively small amount of persistent state. Gat [24] points out that the solution is not to eschew the internal state, but to use it only for modeling information at high levels of abstraction.

Our analysis in this paper is dependent on a crucial observation regarding the temporal structure of purely reactive systems. Control parallelism as in the subsumption architecture is different from temporal parallelism. The conflicts we are addressing are not control conflicts but temporal sequence conflicts. In order to investigate such conflicts, it is necessary to separate the control structures of behaviours, from their temporal structures. While the control structure is essentially parallel, the temporal structure of behaviours is mostly sequential. In practice, one behaviour usually provides the stimulus for another, so that there is often a clear temporal sequence in which behaviours are executed. Thus behaviours may be thought of as occurring in temporal sequences. We model these by a sequentiality operator, and call such sequences *behaviour chains*. These are not to be confused with the control model, which remains largely parallel.

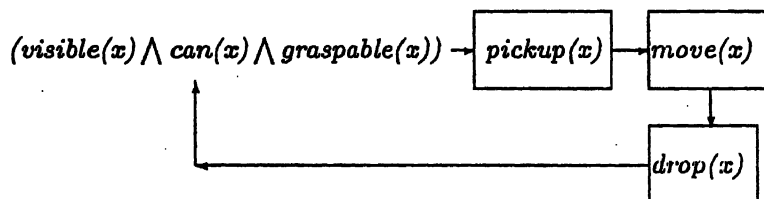


Figure 1. *Conflict in picking and placing the can.*



Arbitration type of conflicts have been envisaged in behaviour systems from the very start. The most common mechanism for dealing with these control conflicts is subsumption [9], which is in essence a prioritization scheme between behaviours that are likely to be triggered simultaneously. However, prioritization methods are inadequate for resolving cyclic conflicts, since the cycle may reappear whenever the suppressing behaviour has completed its execution. In this paper we show that such cycles can be avoided only by modifying the behaviours themselves, and we introduce two such modifications, based on specializing the stimulus or restricting the action of a behaviour. One of the key results of the paper is that any such modification reduces the usefulness of the behaviour structure and makes it less flexible.

Next we describe the models of behaviour and our notation for behaviours in *Chapter 2*. In subsequent chapters we deal with the detection of cycles in behaviour chains, and our strategy of behaviour refinement for eliminating the conflicts. Similarities between AI planning and the behaviour based approach are discussed in *Chapter 5*. *Chapter 6* discusses alternate architectures for robot control and role of learning and behaviour modification. *Chapter 7* presents our conclusions.



Chapter 2. What Is a Behaviour?

AI researchers, psychologists, cognitive scientists, ethologists and roboticists all use the term behaviour in senses that are related but are fundamentally different.

At one end of the spectrum is Brooks [9] who looks upon behaviours as a type of intelligent module, an input-output relation to solve small problems. Hopefully these can be combined to solve larger problems. Each behaviour is affected by and affects only the external world, except for the *inhibition* and *suppression* control signals which disable the output from other behaviour. Brooks reports that inhibition and suppression are the mechanisms by which conflict resolution between actuator commands from different layers is achieved [12]. Conflict resolution tends to happen more at the motor command level, rather than the sensor or perception level [11]. But this is true for resource sharing type of conflicts and does not hold for cyclic conflicts. There is no other form of communication between modules, in particular there is no shared global memory. The stimulus to a behaviour is boolean, and is tested by an *applicability predicate*. A *transfer function* defines what output to generate for the given input. This is the model of behaviour investigated in this paper.

Minsky [43] suggests thinking about goal directed behaviour as an output of a difference engine that measures the differences between the world state and the goal state and then takes actions to reduce these differences. This sounds like planning. On the other hand, Simon [48] feels that complex behaviour need not necessarily be a product of an extremely complex system, rather, complex behaviour may simply be the reflection of a complex environment. Ethologists like Gould emphasize the importance of motivation in addition to sensors and effectors [26].

A number of reactive models have been used in robot navigation tasks. The well known model



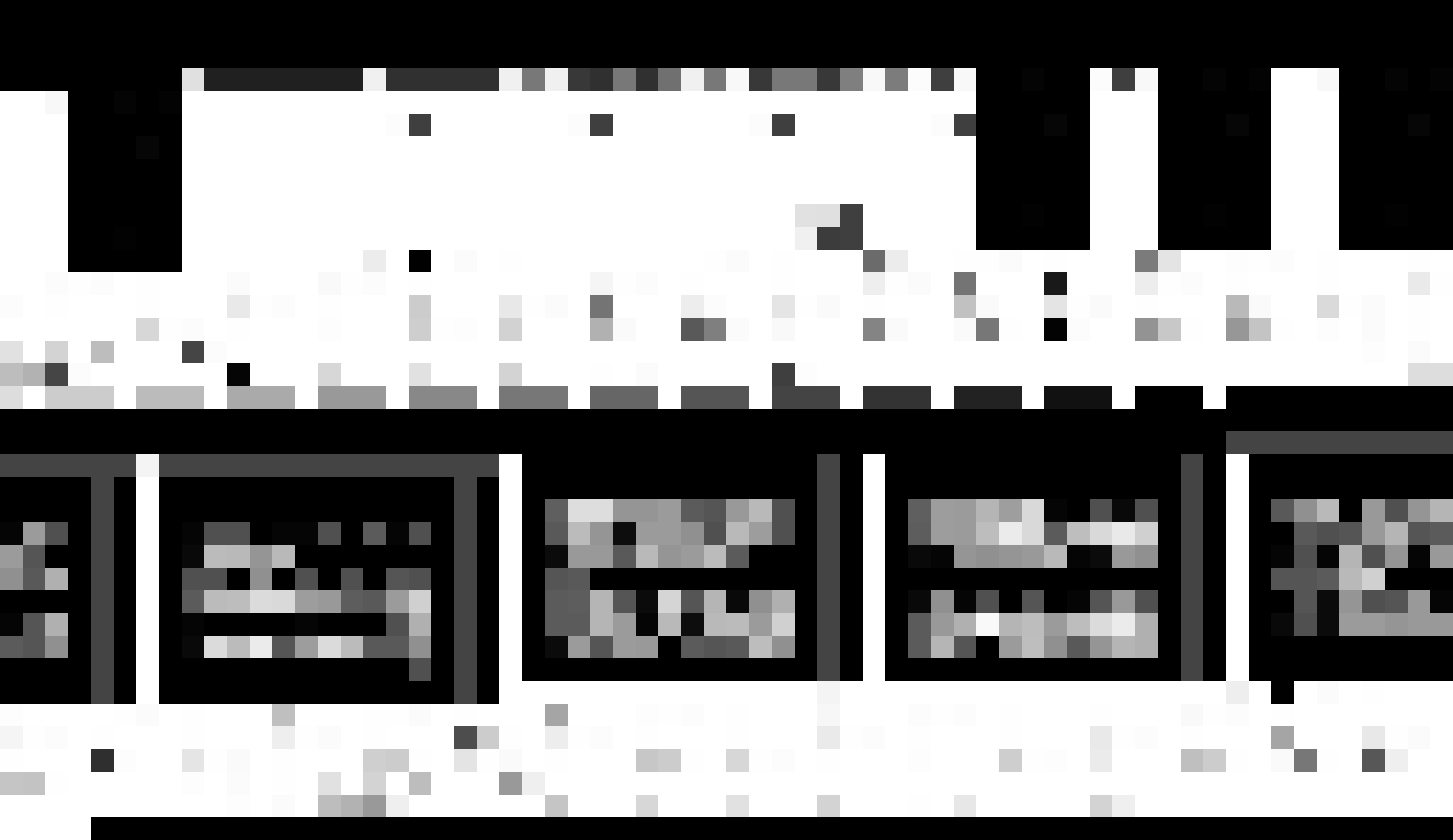
by Arkin proposes *motor schema* as a basic unit of behaviour specification for the navigation of a mobile robot [2]. Motor schemas like *move-ahead*, *avoid-static-obstacle*, *move-to-goal* are used for robot navigation. Potential fields are used to guide operation of the schemas and there is a notion of stimulus strength in such models. However, many such models are not purely reactive since *a priori* world knowledge may be used to guide the selection of the schemas that are necessary for successful completion of the robot's mission. A schema is a model used in psychology to describe the interaction between perception and action and can be adapted to yield a mobile robot system that is highly sensitive. The word schema is linked with the motor program model used in psychology which stores muscle command sequences for executing complex actions quickly; a schema permits such motions to have some parameters as well, the precise relationship being learned over time in human subjects [47].

2.1 Notation

Models of behaviour vary widely between robotics, psychology, cognitive science. Maes [39] models a behaviour module as a 4-tuple $\langle c, a, d, \alpha \rangle$ which represent pre-condition, add list, delete list and activation level. In this work we have followed the behaviourists and adopted a 3-tuple model of behaviour: stimulus, action, consequence.¹

An elemental behaviour module β takes the form $\langle s, a, c \rangle$. Both s and c are commonly defined in terms of a logical expression. A response is an action, which results in certain changes in the self-state of the robot or in the state of the environment. In our strategy of detection and elimination of conflicts, action is not directly useful, thus we prefer to drop it in examples here though it remains an important part of our notation.

¹Psychologist Blackman [8] uses the notation $A : B : C$, for behaviour where A = antecedent or setting conditions, B = behaviour, C = consequence.



Much of our analysis studies the temporal relations between behaviours. For this purpose we define the *dominant period* of a behaviour as that period when the behaviour is active. This term is analogous to the duration of an event. In most behaviour implementations, behaviours become dominant in a temporal sequence. We use the symbol “:” to denote this, i.e., $\beta_1 : \beta_2$ says that behaviour β_2 becomes dominant following behaviour β_1 . It is read as “precedes”: the module β_1 precedes the module β_2 if and only if there exist time instants $\theta_1, \theta_2, \theta_3, \theta_1 < \theta_2 < \theta_3$ such that at θ_1 , β_1 is dominant but β_2 is not and at θ_2 , β_1 and β_2 are both active or they meet and at θ_3 , β_2 is active but β_1 is not. The *temporal chain* of behaviours $C = \{\beta_1 : \beta_2 : \beta_3 : \dots : \beta_n\}$ is an ordered set where the ordering is based on temporal precedence as defined here. The temporal analysis performed here may be thought of as *a posteriori*, i.e. after the behaviours have been designed, or even after it has executed. However, in practice, it appears that most behaviour designers have in mind a certain task, and this can be performed typically by executing behaviours in a certain sequence, so much of the analysis also applies to behaviour design itself.

2.2 Behaviour Chain

We define a *behaviour chain* as a sequence of behaviour modules $\{\beta_1 : \beta_2 : \beta_3 : \dots : \beta_n\}$ where each module except the first one follows from the earlier one in the chain. Here the action of the earlier module changes the situation in such a way that the newly changed part of the situation is in turn a stimulus for the next module in the sequence. If the consequence and stimulus include a finite universal state as well, then we can say that the stimulus s_{i+1} of the behaviour module β_{i+1} is logically implied by the consequence of the module β_i i.e. $(c_i \Rightarrow s_{i+1})$. Also no other module of higher priority should have a suppressive effect on β_{i+1} . Consider, for example the behaviour *explore* which can be defined in terms of lower level behaviours *go-to-new-loc* that enables the robot to go to



the location of a new object and *go_around* that enables the robot to go around the object to find its extents. Our definition of the behaviour *explore* is, $explore := (go_to_new_loc : go_around)$
 $go_to_new_loc := \langle notvisited(x) \wedge loc(x, L), consequence_of_go_to_new_loc_action \rangle$
 $go_around := \langle newloc(L) \wedge atsafedist(L), consequence_of_go_around_action \rangle$.

This discussion does not handle concurrent execution of behaviours. Consider for example, the task of filling bottles in which the *fill_bottle* module holds the bottle below the tap and *check_level* checks the level of fluid in the bottle while the bottle is being filled. Such behaviours will be “forked” off the chain at some point and will “re-enter” at some other point. The temporal sequences between behaviours in the parallel branches in this case are unclear. Our model can be extended to this case. However, one could decompose concurrent behaviours into individual behaviour chains, e.g. $\{ \beta_1 : \beta_2 : \dots : \beta_m : \beta_l : \dots : \beta_l : \beta_j : \dots : \beta_n \}$ as well as $\{ \beta_1 : \beta_2 \dots \beta_m : \beta_k \dots : \beta_p : \beta_j \dots : \beta_n \}$ (Figure 2).

We define a *behaviour space* B as a set of behaviour modules. A temporal chain of behaviours C is said to be composable from B if and only if $(C = ordered_set \{ \beta_i \} \wedge (\forall i) \beta_i \in B)$. We express this by $C \prec B$, read as C is composable from B . A *stimulus space* Σ of a behaviour space B is the union of all the stimuli of all behaviour modules in B .

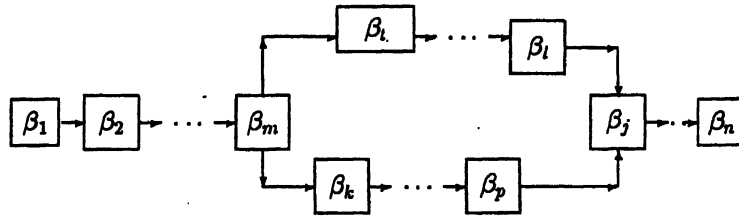
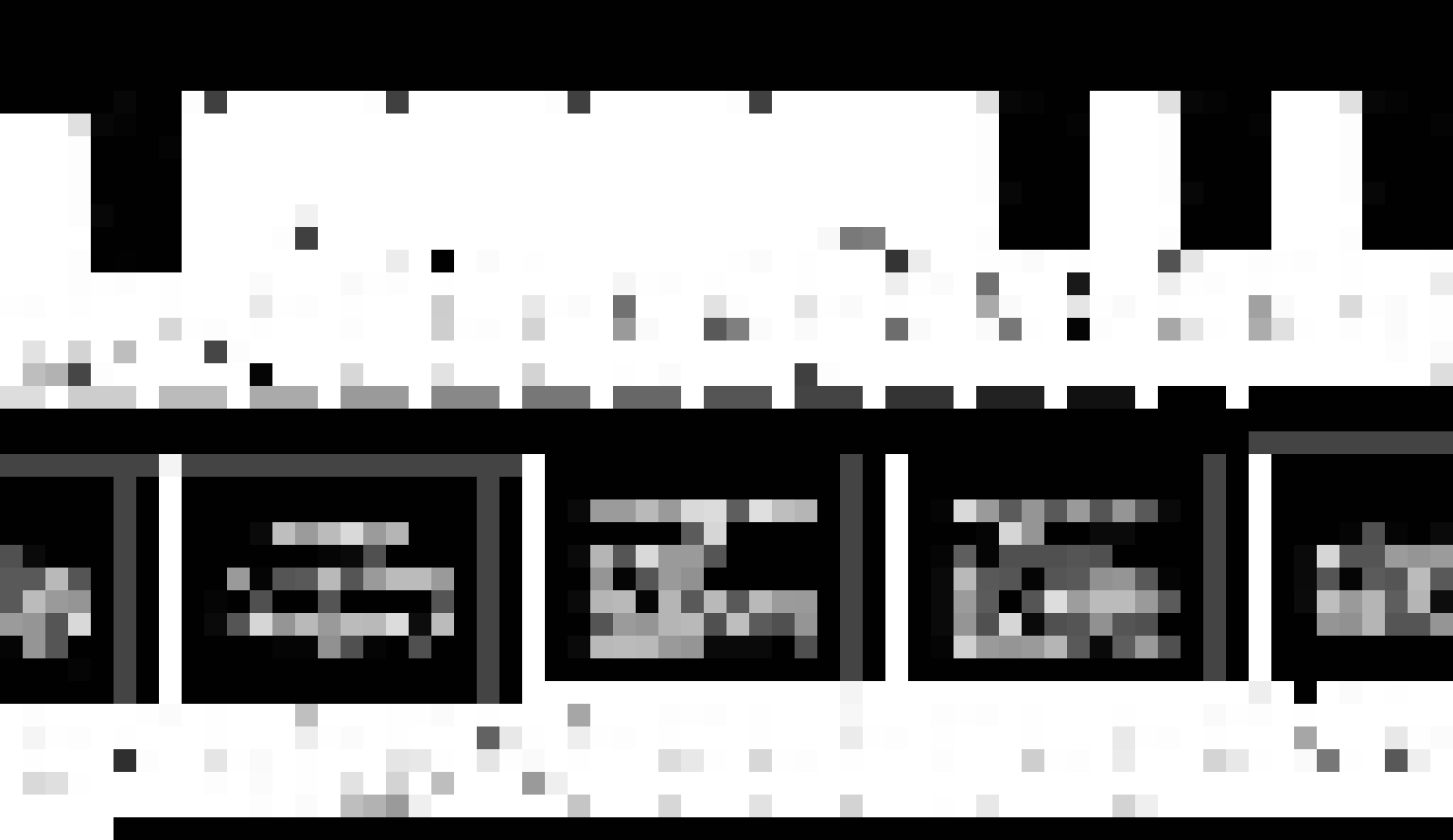


Figure 2. Behaviour chain with concurrent behaviours

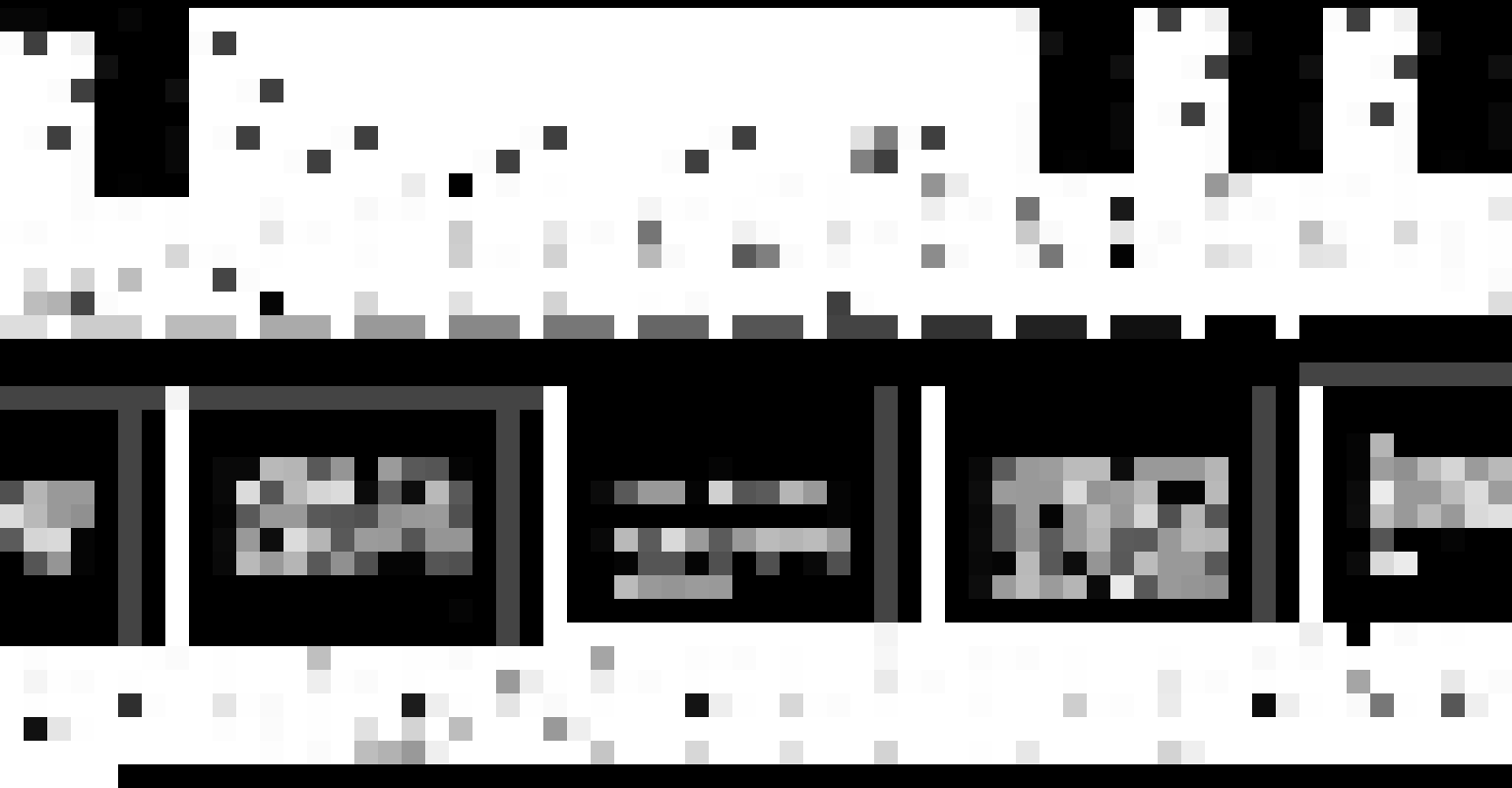
In a behaviour β_i with action a_i , the consequence c_i is the set of predicates affected by the action. e.g. $conseq(pickup_action)$ is $(handholds(x) \wedge can(x))$. As a consequence of action of a mod-



ule, some new predicates become true and some old predicates become false. This is similar to the *add* and *delete* lists in STRIPS [22] (see *Section 5*).

2.3 Frame Problem

This model is subject to the frame problem i.e. considerations of universal truths [25] [28]. In our context, behaviour module β_1 leads to β_2 when $(c_1 \Rightarrow s_2)$. But here we need to consider the universal truths also. Let $Universe = X \wedge Y \wedge Z$ and $c_1 = A$ and $s_2 = X \wedge A$. Then β_1 leads to β_2 but $(c_1 \not\Rightarrow s_2)$. Thus when we say that $(c_1 \Rightarrow s_2)$ we mean that a part of s_2 was true earlier and some literals in c_1 cause the rest of s_2 to come true. Thus $\beta_1 : \beta_2$ is not equivalent to $(c_1 \Rightarrow s_2)$. Some psychologists [36] have modeled this universe as the “situation”, resulting in the *3-tuple* (stimulus, situation, response). It is well established that the stimuli required to activate a complex behaviour pattern are usually only a very small subset of the total amount of sensory information available to the animal, known as *sign stimuli*. The distinction between situation and stimulus is quite unclear in the behaviour models. Also if situation is the same as universe, then it may not be finite. In our model, limiting our analysis to a small set of behaviours, we temporarily overcome this objection by considering a finite behaviour space, where the stimuli and the consequences are enhanced by all universal truths that are affected in any individual behaviour of a chain. While we realize that this definition is different in its implementation from that typically used in reactive behaviour systems, we adopt it since it does not cause any fundamental changes within the behaviour chain, and it allows us to develop the argument skirting the philosophical problems.



2.4 Power, Usefulness and Flexibility of Behaviours

In order to compare different behaviour systems, it is necessary to have some measures of what can be achieved by these systems. Here we define a few relative measures for comparing behaviours.

Definition 1

• Power :

A behaviour $(\beta := \langle s, a, c \rangle)$ is more powerful than $(\beta' := \langle s', a', c' \rangle)$ iff $(s' \Rightarrow s) \wedge (c \Rightarrow c')$

Using the convention that predicate p_1 is stronger than predicate p_2 if $(p_1 \Rightarrow p_2)$, we can say that a behaviour is more powerful if its stimulus is weaker and its consequence is stronger. In other words, it can be triggered at least as frequently as a less powerful behaviour and results in at least as strong a consequence. For example, a general behaviour *pick-up* that can pick up any object is more powerful than *pick-up-coffee-cup*, since the stimulus for the latter is stronger, but the consequences remain the same. A behaviour space B is more powerful than the behaviour space B' if B' can be obtained from B by replacing some module $\beta \in B$ by less powerful module β' . Thus the term “powerful” can apply to both individual behaviours *and* an entire behaviour space, but we trust this overloading will not be confusing.

• Usefulness :

A set of behaviours is more useful if it can perform more tasks. A behaviour space B spans the task space τ if and only if $\{ \forall (t \in \tau) (\exists (C \prec B) \text{ fulfills } (C, t)) \}$. The *greatest fulfillable task space* $\tau_G(B)$ is the largest task space that is spanned by the behaviour space B . The *usefulness* of a behaviour space is defined as the ratio $\frac{|\tau_G(B)|}{|B|}$. Thus, a behaviour space is more useful if its cardinality is lower (it has fewer behaviours), but it can fulfill as many tasks as some other behaviour space with more behaviour modules.



- **Flexibility :**

A behaviour space B is at least as flexible as behaviour space B' if $\{\forall t \in (\tau_G(B) \cap \tau_G(B')), \exists (C \prec B) \mid \{ \text{fulfills}(C, t) \wedge \forall (C' \prec B') \{ \text{fulfills}(C', t) \Rightarrow |C| \leq |C'| \} \}\}$. In other words, any chain composable from B' is at least as long as a comparable chain in B that performs the same task.



Chapter 3. Detection of Conflicts

Any behaviour chain leading to non-fulfillment of the desired objectives can be said to have a conflict. Let a chain $C = \{\beta_1 : \beta_2 : \dots : \beta_n\}$ be the desirable behaviour sequence that achieves a desirable outcome. There are three types of conflicts that can cause the chain C from not being executed, by breaking the sequence $\beta_i : \beta_{i+1}$.

Definition 2

(a) **Extraneous behaviour Conflict:** $\beta_i : \beta', \beta' \notin C$, an extraneous behaviour β' is triggered by β_i , this can lead to irrelevant consequences that are undesirable. This type of conflict can often be foreseen by the designer and controlled by suppressive link. (Mcfarland [41] defines conflict as a state of motivation in which tendencies to perform more than one activity are simultaneously expressed. We find similarity between this definition and our definition of extraneous behaviour conflict.)

(b) **Cyclic Conflict:** $\beta_i : \beta_k, \beta_k \in C, k \leq i$, this is a cyclic conflict, discussed in detail later in this chapter.

(c) **Skipping Conflict:** $\beta_i : \beta_k, \beta_k \in C, k > (i + 1)$, this is a skipping conflict and may not achieve the full task. Such a situation causes many of the intermediate behaviours to be skipped. Thus some of the stimuli required for the behaviours β_{k+1}, β_{k+2} etc. may not have become available. This type of conflict can be treated in a manner analogous to extraneous behaviour conflicts. The type of behaviour that we are investigating is the *cyclic conflict*, where, both β_{i+1} and β_k may be triggered and clearly the triggering of β_k would lead to a cycle (Figure



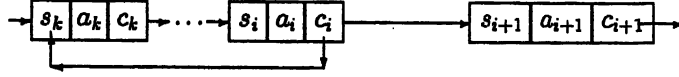
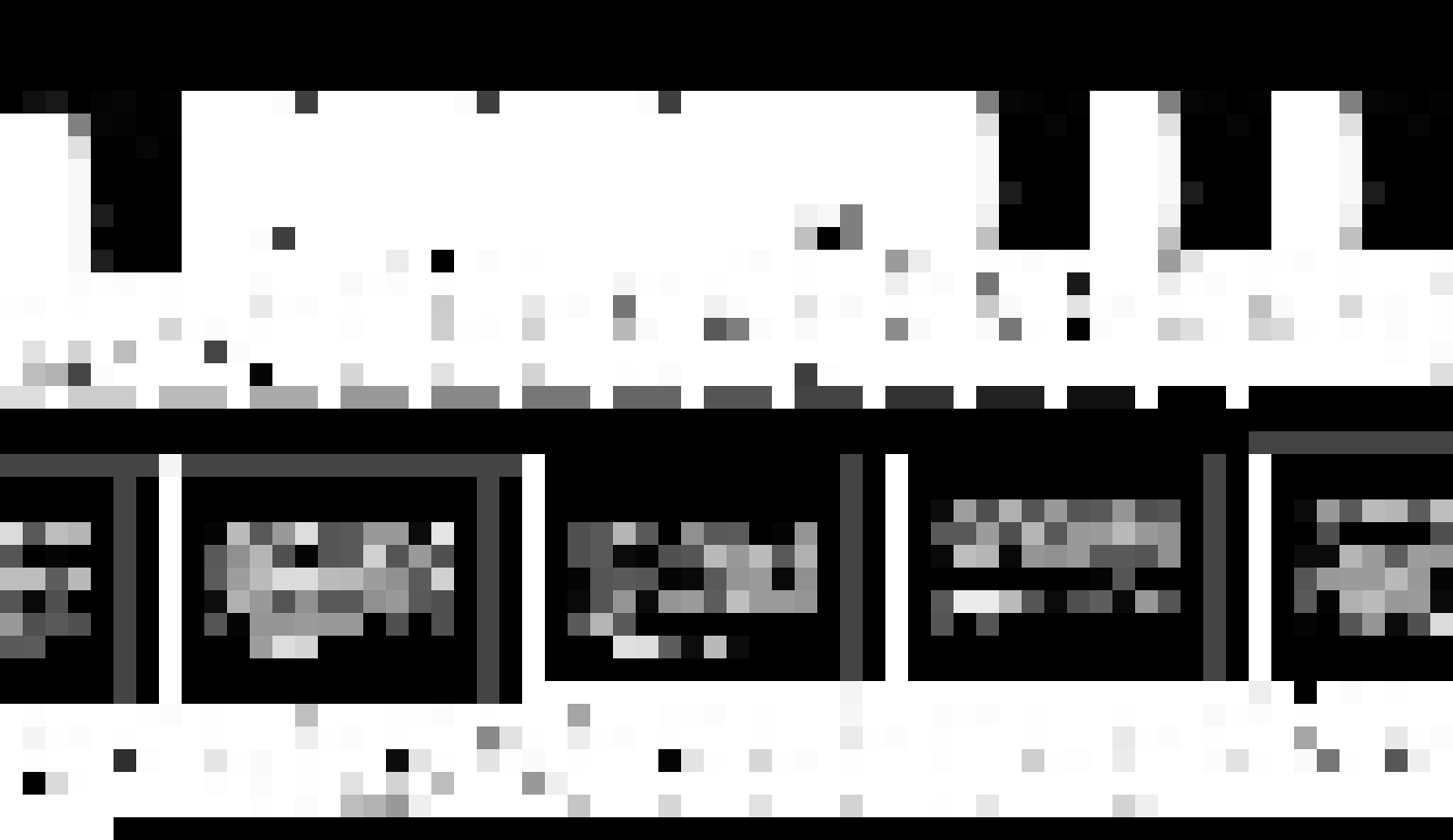


Figure 3. *Cyclic conflict in a temporal chain of behaviours.*

3).

3.1 Terminated Cycles

All cycles need not result in a conflict. Let us say that we have a large block of butter that needs to be cut into a number of smaller pieces. A module *cut* achieves it. *Cut* gets activated when a block of butter is visible and is cuttable by the robot and its action is the division of the piece of butter into two halves. The cycle of cutting the successive smaller pieces of butter, is indirectly implementing a recursive behaviour. We specify the smallest acceptable size of the piece of butter by specifying the limit λ , which introduces the termination condition (Figure 4). The behaviour *cut* has a cycle in it, but this is not a conflict situation since there is a termination condition with the characteristic that after a finite number of cycles, the consequence of the behaviour itself causes its stimulus to become false. Terminated cycles may also involve more than one behaviour; here the chain contains a loop but the consequence of the last behaviour c_n in the chain eventually causes the stimulus of the first s_1 to become false. All such terminated cycles can be detected by a test to see if some literal γ common to c_n and s_1 , or its parameters, are changing at the end of each cycle such that eventually γ becomes false in c_n , thus negating the implication. As soon as a cycle is detected in a behaviour chain, the literals in θ , the most general substring between c_n , s_1 are checked. Typically, the parameters involved in the predicate (like b and λ above) should be investigated to see if this is the case. A more powerful termination detection can be performed if the predicates are not boolean but have a strength of stimulus characteristic, in which case a



reducing strength is indicative of possible termination.

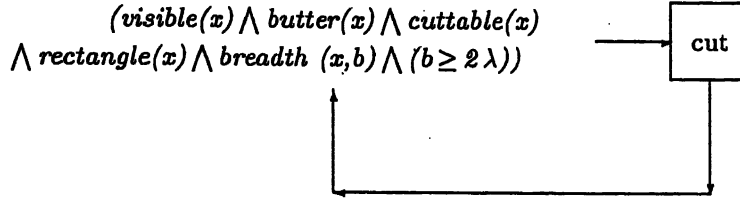


Figure 4. *Termination condition in a recursive cycle. $(b \geq 2\lambda)$ is the termination condition.*

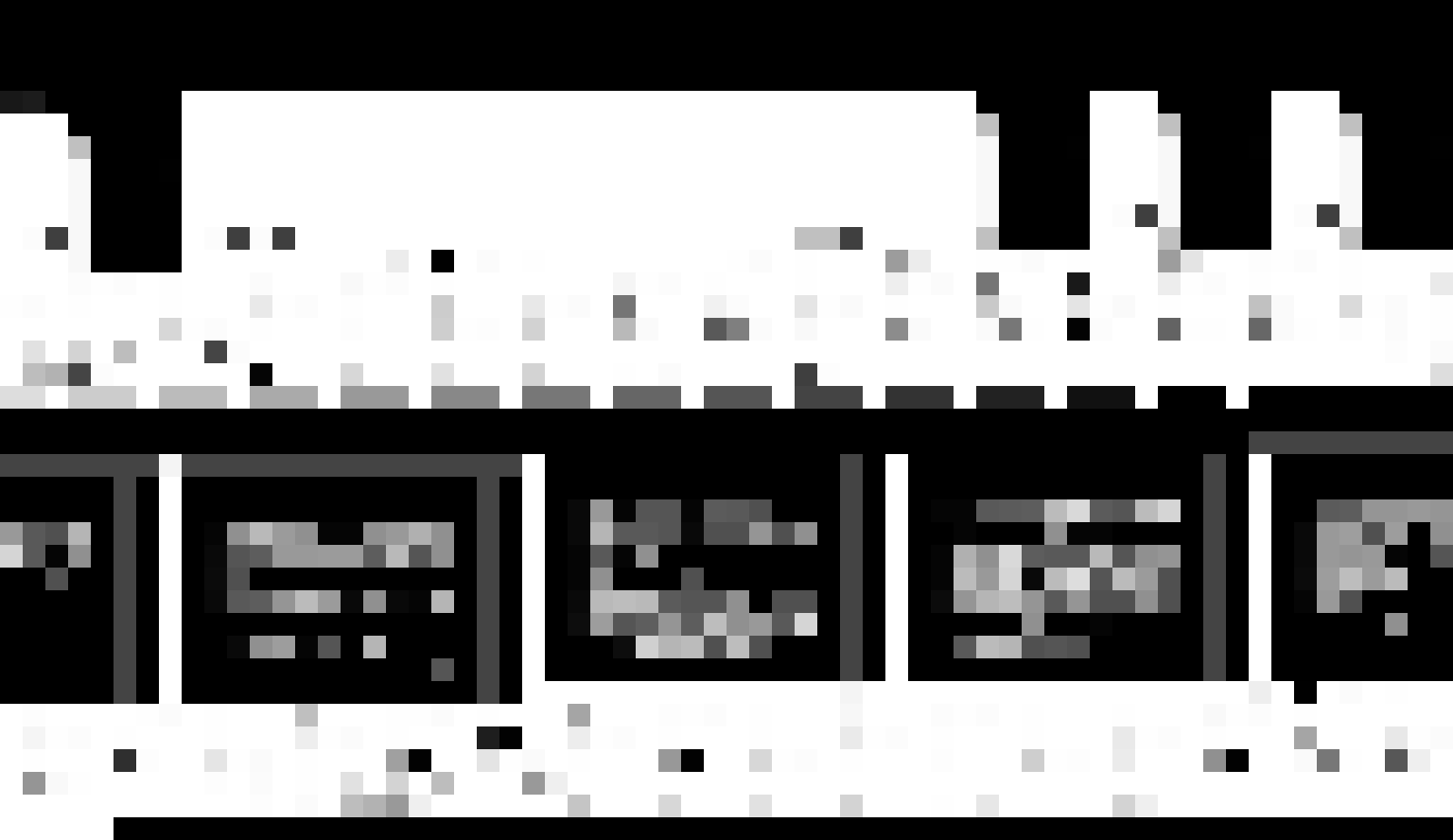
3.2 Prioritization

We consider the role of suppression, inhibition and delayed action in conflict resolution. If $\alpha : \beta$ is a possible but undesirable behaviour sequence, these are the mechanisms to modify it. Unfortunately, there does not appear to be a clear definition of these terms in the behaviour literature. For our purposes, we consider four cases :

(1) *Simple temporal sequence* : This is $\alpha : \beta$, which means that behaviour module α precedes the module β .

(2) *Suppression* : Consider $\alpha : \beta$ such that the output of β is suppressed by γ . After α has executed, stimuli for both β and γ are available. The action of β does not take place. Then the chain $\alpha : \beta$ is broken and $\alpha : \gamma$ occurs.

(3) *Inhibition* : Consider $\alpha : \beta$ such that the output of β is inhibited by the output of γ . In this case the action of β takes place, but only after γ is no more dominant. Here the chain $\alpha : \beta$ has the module γ inserted, so we have $\alpha : \gamma : \beta$. This assumes that the consequence of γ does not destroy the stimulus for β . Inhibitive links temporarily forbid the robot from attempting a certain operation without disturbing the rest of its control system, e.g. if the robot notices a new spot while wandering, then the output of wander is inhibited by the output of module from explore level, thus after the spot has been explored, wandering will again continue.



(4) *Delayed action*: This is a special case of inhibition, where the inhibitive link remains effective for some time t_{delay} even after the inhibitive module is no longer dominant. Connell [18] uses the term *retriggerable monostable* to capture this sense. If $\alpha : \beta : \delta$ is the chain in which the output of β is inhibited by γ , then after α , γ executes and remains dominant for t_{delay} after its stimulus s_γ is no longer available, then if stimulus for β is still available, β executes else δ executes. This action causes the subsequent behaviour to be delayed. This is similar to Skinner's law of after-discharge which states that the response may persist for some time even after the stimulus has vanished.

These three measures establish the relative priority between behaviour modules. In practice, behaviour system designers use them to obtain a desirable behaviour chain that meets the current task objectives. Thus the reasoning leading to these prioritization lies at the motivation level and not at the reactive level. This is clearly a meta-level process, and some form of global knowledge representation will be needed in order to reason about these links from within the behaviour system. While it is possible to detect conflicts, installing priorities in purely reactive systems therefore remains a prerogative of the designer. In the can example, had the behaviour system used the context i.e. the fact that the can was not to be picked up, the can would not have been picked. This challenges the assumption of locality of knowledge which claims that currently obtained knowledge about the world through perception is sufficient for doing tasks.

What type of conflicts can be avoided by prioritization? Clearly, extraneous behaviour and skipping type of conflict can be avoided. If the robot has behaviours *grab_book* and *grab_can* and it simultaneously sees both of them, then that conflict is of extraneous behaviour type if the robot has only one gripper. This too can be solved by adding a suppressive link. However cyclic conflicts $\beta_i : \beta_k, k \leq i$ cannot be reliably eliminated by these prioritization schemes. Inhibition cannot eliminate cyclic conflicts since the offending module β_k is executed immediately following



the inhibiting module. There can be very few cyclic conflicts which can be resolved permanently by retriggerable monostable. This is similar to *behavioural hysteresis* discussed by Beer *et al* [7] where the *edge following* behaviour of an insect persists for a short period of time even after the sensory stimulus which initially triggered has been removed. As another example of the effect of delay, consider a module *move_cradle* that is triggered when the robot detects the cry of a child, $move_cradle := \langle (cries(x) \wedge child(x) \wedge incradle(x)), consequence_of_move_cradle_action \rangle$.

After the child cries in the cradle, robot starts moving it and child stops crying. Since the stimulus has vanished, the robot no more moves the cradle. But as the motion of the cradle has stopped, the child starts crying again (Figure 5). The robot again starts moving the cradle. But had the robot moved the cradle for a little more time even after the child had stopped crying for the first time, i.e had the behaviour module *move_cradle* remained active for a longer time, say upto t_2 even after the stimulus of cry of the child had vanished, the child would have stopped crying and cycles would have been avoided. This can be achieved by a retriggerable monostable. However, note that the length of the delay, the type of behaviour where this will work, the probability of achieving success are all not predictable here. For this to work, it requires that the behaviour that is delayed has a higher priority over the behaviour which leads to cycles. Moreover, during the delay period, it assumes that some intervening action (e.g. wander) causes the stimulus leading to the cycle to vanish. The delayed action scheme assumes the existence of some intermediate behaviour β' such that the consequence of β' invalidates the stimulus of β_k (in Connell [18], this β' is available in the form of *wander* which is likely to make the dropped can temporarily invisible); but β_k may still be triggered after time t_{delay} . This is clearly an ad-hoc measure and cannot be guaranteed to work. Even Brooks in some early implementations, uses time lags along with the inhibition signal, but these were not used in the later models.



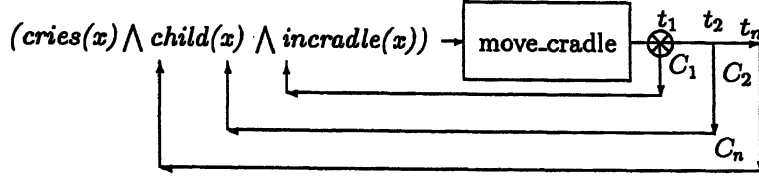


Figure 5. A conflict that can be prevented by retriggerable monostable

Suppressive links are not guaranteed to kill the stimulus of β_k , hence β_k may be active after dominant period of the suppressing module is over.

Thus within the scope of the three prioritization schemes discussed here, it is not possible to guarantee that cyclic conflicts will be avoided. But before we consider our approaches to cyclic conflict resolution, we must first be able to identify cyclic conflicts.

3.3 Cyclic Conflicts and Behaviour Graphs

A *digraph* G is a graph consisting of a set V of points and a set E of ordered pairs of distinct points. Any such pair (u, v) is called an *arc* or a *directed line* and will be denoted by uv . A *walk* of the graph G is an alternating sequence of points and lines, where each line is incident with the two points immediately preceding and following it and the first and the last entry are points. A walk joining β_1 and β_n may be denoted by $\beta_1 : \dots : \beta_n$. A closed walk has the same first and last points.

Each behaviour module can be represented by a node in the graph and a directed edge between two nodes represents the temporal sequence and the fact that the consequence of the earlier module implies stimulus for the latter module. We see that our graphical representation of behaviour modules is a digraph as per the definition stated above.



Lemma 1(a). Whenever there is a cyclic conflict, there is a cycle in the temporal graph of behaviours.

Proof :- A cyclic conflict arises when there exists module β_i such that its consequence implies a stimulus of module β_k that is before β_i or is β_i itself. In the temporal graph therefore, there exists a link from β_i to β_k , and also a link, in the normal triggering sequence, from β_k to β_i . Hence there is a closed walk in the temporal graph, hence a cycle. \square

Lemma 1(b). Whenever there is a cycle in the temporal graph of behaviours that is not terminated by a recursive condition, there is a cyclic conflict.

Proof :- A link from β_i to β_k in the temporal graph of behaviours signifies that β_k is triggerable after β_i . This can break because of inherent termination condition (recursive termination). In all other cases β_k will be triggered after β_i .

Consider the behaviour chain C which is found to have a cycle in its temporal graph. Let the walk corresponding to this cycle involve the sequence β_k, \dots, β_i in the ordering as per the chain C . However, since this is a walk, there must be a link coming into β_k from some β_i which is the same as β_k or occurs later in the chain i.e. $k \leq i$. Since this walk is not terminated recursively, this means that there is a cyclic conflict in accordance with the definition of cyclic conflict. \square

3.4 Conflict Detection Algorithm

Kube & Zhang report that detecting cyclic conflicts remains an interesting challenge [33]. Consider the temporal behaviour sequence shown as a graph (Figure 6). The creation of this graphical representation is guided by the action sequences in a behaviour. Various cycles C_1, C_2, C_3, C_4, C_5 are shown. In case of a cyclic behaviour conflict, either the entire graph or a subgraph of it is a cycle. Detecting cycles in a graph is a standard problem. Breadth first search can be used



to detect cycles in $O(n^2)$. If an old node is revisited in the search, this implies a cycle. Cycles can also be detected in powers of the adjacency matrix- the (i,j) entry in X^r equals the number of different, directed edge sequences of r edges from the vertex i to vertex j [20]. If not a recursive cycle, we can eliminate the cycles using *behaviour refinement* discussed in *Chapter 4*.

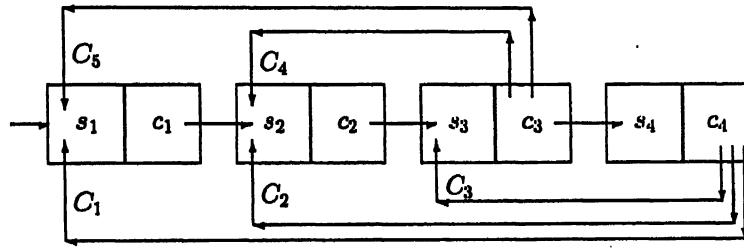


Figure 6. *Graphical representation of the modules and the cycles in it.*

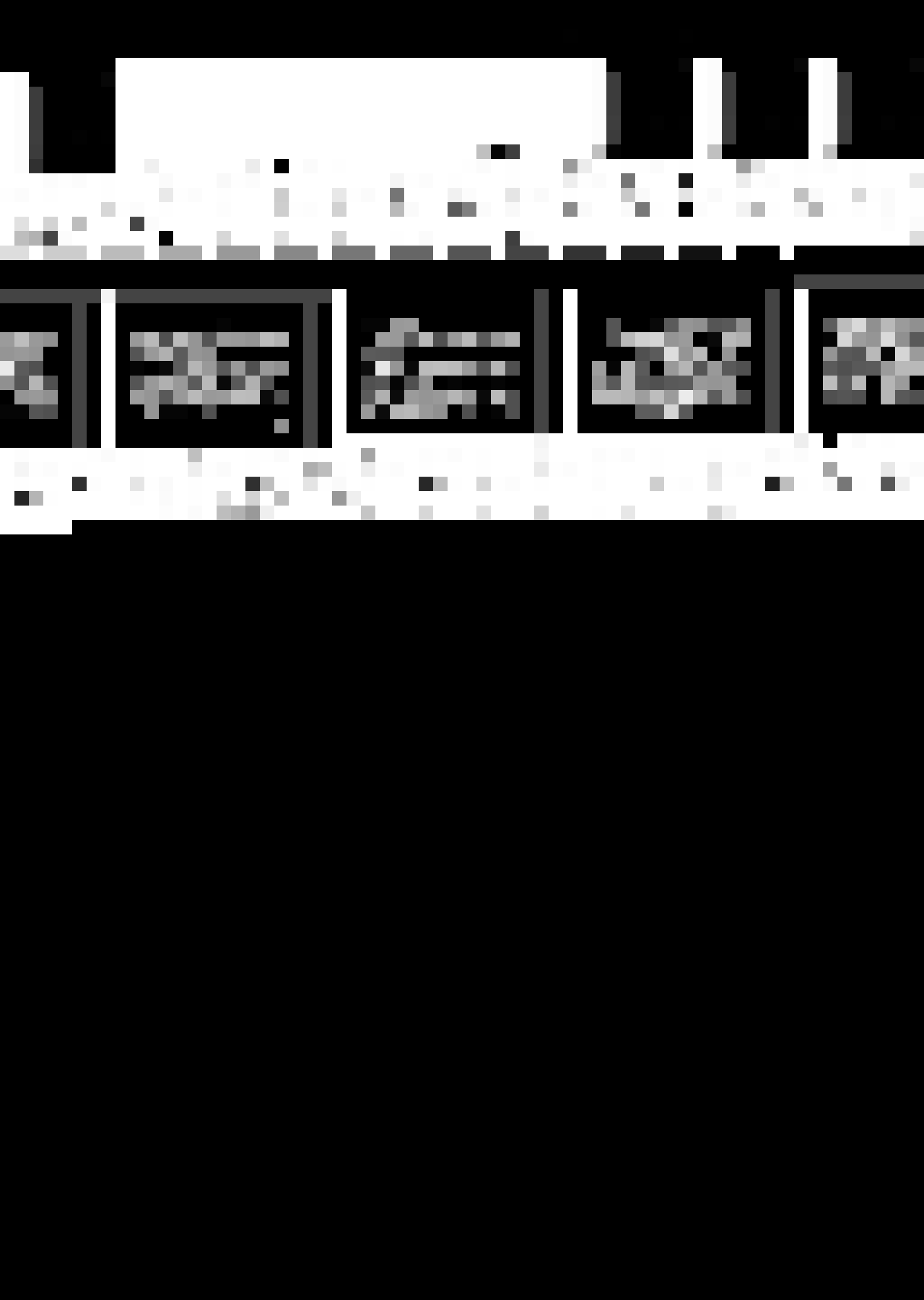


Chapter 4. Behaviour Refinement

The method suggested by Brooks for avoiding cycles is to manually introduce suppressive or inhibitive links at the time of defining the system. These methods have the advantage that the behaviour module itself is not modified. As shown in section 3.2, these links do not always eliminate cycles, since the intervening action may not have negated the stimulus for the offending behaviour. Hence the cycle can start again after the higher priority behaviour stops being dominant. Is there any method where the behaviours themselves can be modified to avoid cycles? We find the answer here. An approach would be to find a small set of directed edges whose removal will render the given digraph G acyclic. Such a smallest set of edges is called as a *minimum feedback arc set* [34]. But the edges that are recommended for removal cannot disturb the triggering sequence, so only backward edges from β_i to β_k , $k \leq i$ should be removed. To do this, we need to nullify $(c_i \Rightarrow s_k)$. Let us explore some behaviour modification approaches that would achieve this.

In psychology, a concept of *behaviour generalization* has been discussed which involves stimulus generalization, situation generalization and response generalization [36]. However as the environment triggers the actions of a behaviour-based robot, we consider stimulus generalization and the situation generalization as one and the same.

Stimulus generalization in the human context has also been discussed [31], for a child afraid of a rabbit as well as the beard of Santa Claus, a generalized stimulus is likely to be any moving object that contains a concentration of white hair. But our definition of a behaviour (section 2.1) may not be same as the semantic meaning of a behaviour in psychology. We look upon stimulus generalization as a means of increasing the power of behaviours, e.g. a robot that picks up cans, should also pick up books, this can be expressed by introducing internal disjunctions



in the stimulus. Kirsh points out that unless robots can generalize stimuli, they will have to be reprogrammed to perform what are essentially the same tasks on slightly different objects. If a robot can pick up a soda can, it should be able to pick up a coffee cup [32]. Since different tasks can now be done by the same behaviour, this increases usefulness. Thus stimulus generalization is one of the objectives of behaviour design.

However sometimes stimulus may be overgeneralized. This leads to conflicts which have to be resolved by achieving an adequate level of generalization i.e. by specializing the stimulus. Cyclical behaviour in wandering and wall following has been reported by [15]. *Stimulus overgeneralization* is the cause of cyclic behaviour in all these cases. It also gives insights into designing cycle eliminating mechanisms.

For the behaviour designer, to set the right level of generalization for the stimuli is not an easy task. Also when we already have a definition of the stimulus and there is a conflict, we need to specialize the stimulus. At the same time, restricting the stimulus too much would lead to task under-performance. For the can pickup example, the stimulus for *pick-up* can be modified (along with an internal state, memory) so that it picks up only those objects it has not seen before; thus using it you can sip your coffee only once. A separate behaviour *e.g. multiple-pick-up* is needed for drinking coffee. This would be a recursive behaviour with a termination condition when the level of coffee is too low. Thus the two new behaviours are specializations of the old behaviour. This is similar to the splitting of original behaviour and the division of goals into *once-only* goals that have to be achieved only once and *permanent* goals that have to be achieved continuously as proposed by [39]. However this leads to increase in the number of behaviour modules and reduces the power and flexibility of the behaviour system.

Whenever there is a cycle in a behaviour chain $C = \{\beta_1 : \beta_2 : \dots : \beta_k : \dots : \beta_i : \beta_{i+1} \dots : \beta_n\}$,



there must be a triggering of the kind $\beta_i : \beta_k$, where $k \leq i$. Then both $\beta_i : \beta_{i+1}$ and $\beta_i : \beta_k$ are possible here. Our objective is to break the $\beta_i : \beta_k$ link without disturbing the $\beta_i : \beta_{i+1}$ or $\beta_{k-1} : \beta_k$ triggerings which are essential to the successful execution of the chain.

- $\beta_i : \beta_{i+1}$ means $(c_i \Rightarrow s_{i+1})$
- $\beta_i : \beta_k$ means $(c_i \Rightarrow s_k)$
- $\beta_{k-1} : \beta_k$ means $(c_{k-1} \Rightarrow s_k)$

Thus $(c_i \Rightarrow s_{i+1})$ must remain true whereas $(c_i \Rightarrow s_k)$ must be made false. In stimulus specialization, this is achieved by specializing s_k and in response generalization, it is achieved by generalizing c_i .

4.1 Stimulus Specialization

Let us consider the conflict in picking up the soda cans, where the freshly deposited can is picked up. If we were to add the condition “not-deposited-just-now (x)” to the stimulus predicate for pickup, then we would only need a small recency memory (recently dropped can). Thus the stimulus for pickup becomes more specialized. However, in doing this, one must be careful so as not to disturb the rest of the chain, i.e. $(c_{k-1} \Rightarrow s_k)$ should still hold but $(c_i \Rightarrow s_k)$ must be broken. Clearly this will not be possible where $(c_i \Rightarrow c_{k-1})$, then any changes we make in s_k such that $\neg(c_i \Rightarrow s_k)$ will also result in $\neg(c_{k-1} \Rightarrow s_k)$. Thus stimulus specialization can be used only if $(c_i \Rightarrow c_{k-1})$ is not true. One model for this is to say that there must be a literal γ such that $(c_{k-1} \Rightarrow s_k \wedge \gamma)$ but $\neg(c_i \Rightarrow s_k \wedge \gamma)$. The conjunction of all such literals $\Gamma = (\gamma_1 \wedge \gamma_2 \wedge \dots \gamma_m)$ is called the maximal difference between c_{k-1} and c_i .

Stimulus specialization works only when $\Gamma \neq \Phi$. Here s_k is modified to $(s_k \wedge \gamma)$, $\gamma \in \Gamma$. It is advisable not to specialize it more than necessary (e.g. by adding more than one literal), since this adversely affects the power of the behaviour. A simpler understanding of the process is obtained



if both c_i and c_{k-1} are in conjunctive form. Then Γ is nothing but the difference $(c_{k-1} - c_i)$ and s_k is modified by conjunction with one of the literals that is in c_{k-1} but not in c_i . Note that since the stimulus is specialized, any stimuli that are required by the action are still available to it.

4.2 Response Generalization

Here the action is modified so that the consequence of the action is weaker i.e. if the old consequence was c and the new one is c' then $(c \Rightarrow c')$ but $\neg(c' \Rightarrow c)$. If c is in conjunctive form, then this implies a reduced length, e.g. we can modify the action of the module *drop* so that while dropping the can on the ground the robot puts it in an inverted position which prevents the robot from detecting that the object is a can. The original consequence was $(visible(x) \wedge can(x) \wedge graspable(x))$ and the modified consequence is $(visible(x) \wedge graspable(x))$. If we decide to modify the consequence by covering the can by something to make the predicate $can(x)$ false, then this leads to addition of a new behaviour module or modifying the action part of the original module, both of which require considerable re-programming, and are expensive. In response generalization, $\neg(s_{i+1} \Rightarrow s_k)$. One way of modeling this would be to say that there must exist σ s.t. $(c_i \vee \sigma) \Rightarrow s_{i+1}$ but $\neg(c_i \vee \sigma) \Rightarrow s_k$. The disjunction of all such σ 's is Σ .

Again, if s_k and s_{i+1} are in conjunctive form, then a simpler understanding is obtained, since $\Sigma = \sim(s_k - s_{i+1})$ i.e. the negation of all the conjunctions that appear in s_k and not in s_{i+1} . This negation is a disjunction of many negative literals $(\sim \gamma)$. In this instance, modifying c_i to $(c_i \vee \sigma)$, where σ may be the negation of a literal γ already appearing in c_i , is understood better as $(c_i - \gamma)$. Since stimuli/consequences are often conjunctive, this difference notion is a useful concept in practice. Thus c_i is modified to $(c_i - \gamma)$, where $\gamma \in (s_k - s_{i+1})$.

Stimulus specialization is easier to do than response generalization, as response generalization



requires that the action should be modified. However, stimulus specialization may not always be possible; e.g. with “not-deposited-just-now(x)” the robot may still pick up an older deposited can. Better solutions to this, such as “never-deposited-before(x)” or “not-at-depository(x)” would require considerable global memory. Therefore, stimulus specialization, while cheaper to implement, may not be available in many instances, since the actions require a minimum stimulus, and specializing it without memory may be counter-productive.

Summary :

• Stimulus Specialization :

When $\neg (c_i \Rightarrow c_{k-1})$.

What : $s_k \leftarrow (s_k \wedge \gamma), \gamma \in \Gamma$

Result : $(c_i \Rightarrow s_{i+1})$ but $\neg (c_i \Rightarrow s_k)$.

• Response Generalization :

When $\neg (s_{i+1} \Rightarrow s_k)$.

what : $c_i \leftarrow (c_i - \gamma), \gamma \in (s_k - s_{i+1})$

Result : $(c_i \Rightarrow s_{i+1})$ but $\neg (c_i \Rightarrow s_k)$.

4.3 Effects Of Behaviour Refinement

Let us now investigate the effects of stimulus specialization and response generalization.

Lemma 2. Removing a cycle from chain C by stimulus specialization or response generalization cannot introduce a cycle in any other chain C' , that did not have a cycle before.

Proof:- Let β_k be the behaviour that was specialized. Now to introduce cycles when no cycles existed before, some link $\beta' : \beta_k$ must have become possible. This means means $(c' \Rightarrow s_k)$ has become true now. This is not possible since c' is the same and s_k is more specific. Similarly, since



c_k has not been modified, no new links $\beta_k : \beta'$ could have been created, hence no new cycle will be initiated in any other chain C' . Similarly it can be shown that response generalization does not introduce new cycles. \square

What is the effect of these cycle elimination strategies on power, effectiveness of the behaviour system? This is discussed here.

Lemma 3(a). Whenever a behaviour space is modified to eliminate cyclic conflicts by stimulus specialization, its usefulness decreases.

Proof :- Let Σ be the stimulus space and $s \in \Sigma$ and s is a conjunction of its members. Now let s be specialized to s' so that $s' \subset s$. Now tasks or subsequent behaviours requiring the predicates in $(s - s')$ will no longer be attended by B . Thus we need a new behaviour β' for some $s'' \subseteq s$, such that $(s'' \cup s') = s$, so that β and β' together serve the stimulus set s which implies that $|B|$ increases and the usefulness of B decreases. If the new behaviour is not added, the usefulness decreases because of decrease in the greatest fulfillable task space. The notion of this proof can be generalized to non-conjunctive stimuli, where also a similar set of unserved stimuli can be found. \square

Lemma 3(b). Whenever a behaviour space is modified to eliminate cyclic conflicts by response generalization, the flexibility and usefulness of the behaviour space decreases.

Proof :- If the response of β is generalized so that $c' \supset c$. Thus $(c - c')$ is not being performed by β . Hence other new behaviours are needed to complete the tasks requiring $(c - c')$ which increases $|B|$. This implies that the usefulness of B decreases. Also addition of new modules increases the lengths of the chains composed to fulfill these tasks resulting in decrease in flexibility. If this not done then some tasks requiring $(c - c')$ cannot be performed which implies that $|\tau'| < |\tau|$ which again means that the usefulness of the behaviour space decreases. The notion of this proof can be



generalized to non-conjunctive consequences, where also a similar set of unserved stimuli can be found. \square

Let us say that we have a behaviour β whose consequence $c = p \wedge q$ leads to a cycle. If we use response generalization, we may have to design two behaviours β' and β'' such that $c' = q$ and $c'' = p$. If β has a stimulus $s = p \vee q$ which is triggered leading to a cycle and if we use stimulus specialization, we may have to design two more behaviours β' and β'' such that $s' = p$ and $s'' = q$. In any case, the number of behaviours goes up and lengths of behaviour chains will increase. We see that this leads to behaviour splitting. In some cases, especially for response generalization, it may not be possible to design an action such that it will fulfill these conditions.

We observe here that both stimulus specialization and response generalization cause reduction in the power of the behaviour. This brings us to our most important results, which have to do with the power and usefulness of behaviour spaces.

Behaviour Modification Theorem. Given two behaviour spaces B and B' such that B is more powerful than B' (i.e. B' can be obtained from B by replacing some behaviours β of B by the less powerful ones β') then:

(a) The greatest fulfillable task space for behaviour space B' is less than that for B i.e.

$$|\tau_G(B')| \leq |\tau_G(B)|$$

(b) Usefulness of B is greater than that of B' i.e. $\frac{|\tau_G(B)|}{|B|} \geq \frac{|\tau_G(B')|}{|B'|}$.

(c) Probability of a cycle is also greater in B .

Proof (a) :- First, let us consider the case where a single behaviour β has been replaced by the less powerful β' . The set of chains of behaviours composable from a behaviour space represents a tree with initial point corresponding to the availability of the right initial stimulus and each



ode in this tree represents a world state which may correspond to the desired task. The greatest fulfillable task space is proportional to the total size of this tree of behaviour chains. Now, either the behaviour β will have more applicability due to smaller stimulus length as compared to the behaviour β' or the behaviour β will have stronger consequences resulting in more behaviours being triggerable. In terms of the task tree, either β will have more parent nodes, or it will have more children. In either case, the branching factor is higher in B than in β' and the size of the task tree will be as large or larger. Since $|B|$ has not changed, the usefulness of the behaviour space, $\frac{|\tau_G(B)|}{|B|}$ has decreased. This treatment can be extended to multiple instances of replacing a strong behaviour by a weak one. \square

Proof (b) :- Since B' is obtained from B by replacing some behaviours, $|B| = |B'|$. From (a), the result follows. \square

Proof (c) :- Let $\beta_i \in B$ and $\beta'_i \in B'$ be two behaviours s.t. β_i is more powerful than β'_i , i.e. $(s'_i \Rightarrow s_i)$ or s_i is weaker than s'_i . Now consider any chain composable in B and B' of n modules, which differ only in that the module β_i is replaced by β'_i . Now consider all behaviours $\beta_j \in C$, $j \geq i$, with consequence c_j . The probability of a cycle $prob-cycle(B)$ is

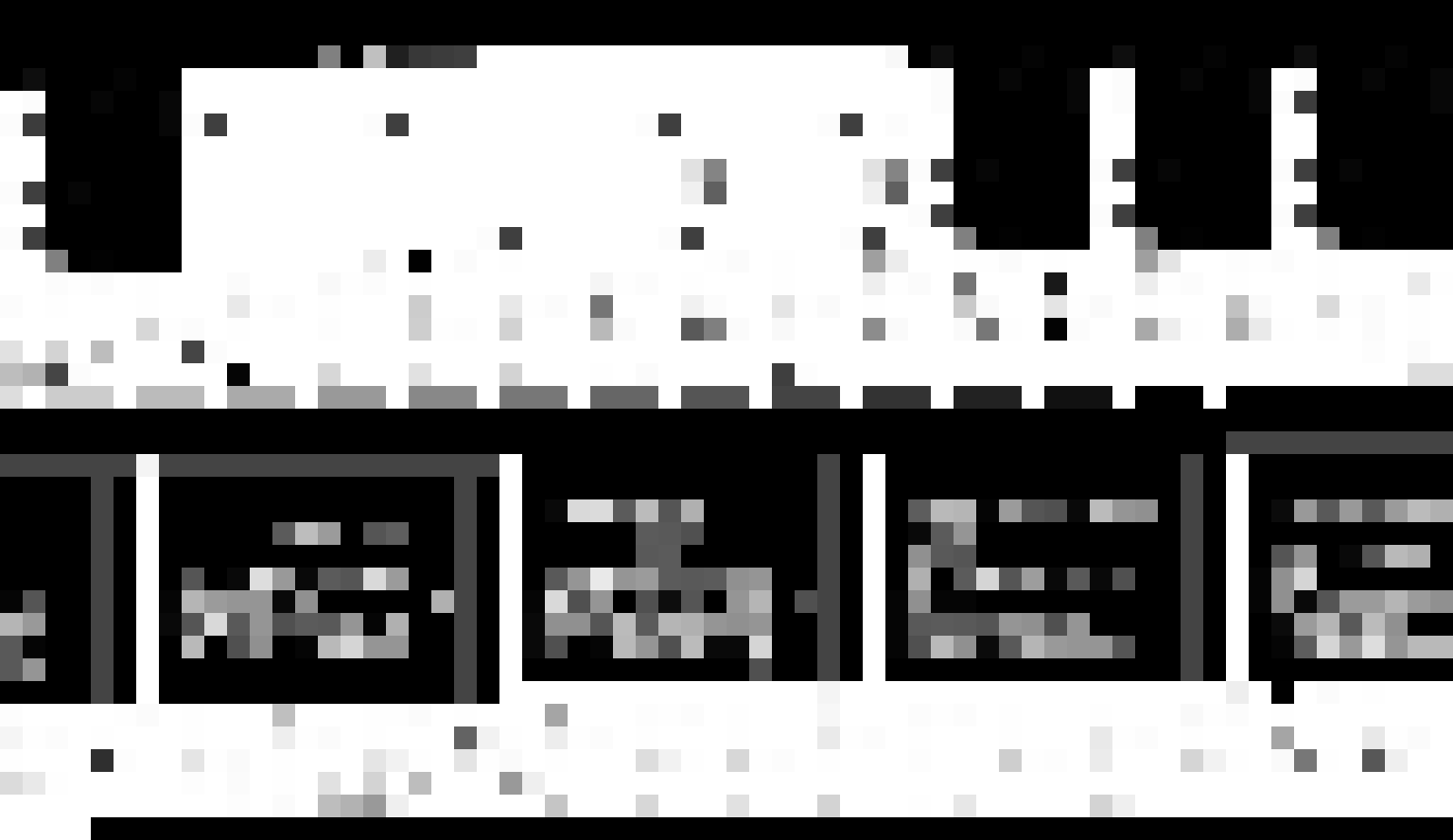
$$\sum_{j \geq i}^n prob(c_j \Rightarrow s_i)$$

and $prob-cycle(B')$ is

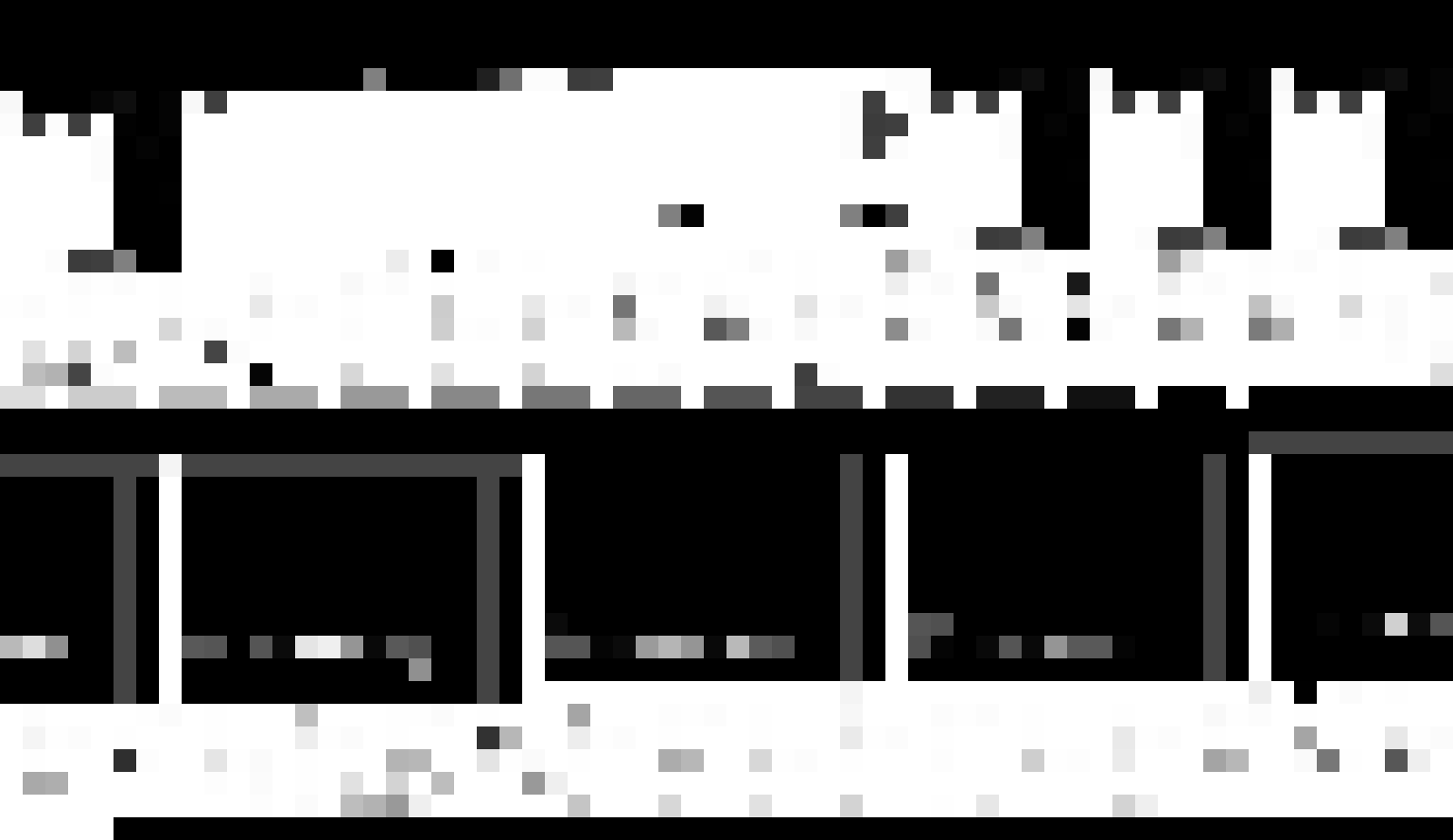
$$\sum_{j \geq i}^n prob(c_j \Rightarrow s'_i)$$

. Clearly, since $(s'_i \Rightarrow s_i)$, $\{\forall j [prob(c_j \Rightarrow s_i) \geq prob(c_j \Rightarrow s'_i)]\}$. Similarly $(c_i \Rightarrow c'_i)$ for which similar analysis can be carried out. Thus $prob-cycle(B) \geq prob-cycle(B')$. \square

Corollary :- If B and B' have the same greatest fulfillable task space τ_G , but $\exists(\beta \in B) \wedge \exists(\beta' \in B')$



single behaviour β_i by β'_i . Then all residuals in the two chains are also identical except R_{i-1} and R_i . If the residuals in C' are stronger, then $(\gamma_i \Rightarrow \gamma'_i)$ and $(\sigma'_{i-1} \Rightarrow \sigma_{i-1})$, i.e. behaviour β is more powerful than β' . Hence by part (c) of the behaviour modification theorem, probability of a cycle is greater in C' than in C'' . The same arguments can be extended for multiple behaviour changes between C' and C'' . \square



Chapter 5. Similarities with AI Planning

In this paper we have focussed on the *temporal* relations between behaviours as opposed to the control relations. This has highlighted an important similarity between behaviour-based modeling and the classical models of planning in AI. The effect of actions, which become new stimuli is similar to the postcondition - precondition structure in means ends planners such as STRIPS [22]. Indeed the similarities go deeper.

Hierarchical planners generate a hierarchy of representations of a plan in which the highest is a simplification or abstraction of the plan and the lowest is a detailed plan sufficient to solve the problem. The approach of assigning criticality to the preconditions of the operators in planning is similar to suppression which prioritizes behaviours. NOAA uses critics or meta-level reasoners. In Arkin's behaviour model [3], a hierarchical planner is used that first chooses appropriate behaviours to minimize the likelihood of occurrence of local minima and local maxima that cause the schema-based navigational methodologies to fail. This is similar to the meta-level reasoners adopted to avoid conflicts in planning. Reflexive planning, local planning, map-based planning and mission planning are used for navigation of a road vehicle [45]. But purists would not consider these to be truly reactive behaviour systems.

A famous example of conflicts in AI planning is the stacker problem discussed in HACKER, a computational model of skill acquisition of Sussman [49]. These conflicts are similar to cyclic conflicts in behaviour models. However, behaviour models differ from planning in some crucial aspects. Locality of behaviour programming makes opportunistic plan-generation automatic, since the relevant behaviour is triggered automatically when stimulus becomes very strong. Also, cycles are much more of a problem in behaviour models since unlike planners, a behaviour does not



“switch-off-and-die” after execution; if the stimulus reappears, it may re-execute, causing a cycle.

Finally, in planning, it is easier to provide for a meta-level reasoning system that resolves conflicts.



Chapter 6. Alternate Architectures

Other solutions for conflict resolution suggest centralized mediation [50], or use of global knowledge [2] which are not in the modular paradigm. Use of real time deadline to detect cyclic behaviour has also been suggested [4] but for a purely reactive agent setting such a deadline is not possible. In particular, it would be impossible to distinguish terminating loops from true cycles. Introduction of new behaviours [18] or general emergency behaviours [27] has also been suggested. In this chapter we discuss alternate architectures of autonomous robots and show how they are not capable of eliminating cyclic conflicts because cyclic conflicts are conflicts occurring at the knowledge level, and not at the representation level. Hence, they are not as easy to detect as extraneous behaviour conflicts. Cycles do not occur at the representation level and hence are harder to solve there. Different architectures differ in their mechanisms for resolving extraneous resource sharing conflicts; however resolving cyclic conflicts has been a difficult problem.

6.1 Potential Fields

The representation of behaviours used, with the first order predicate calculus notation for stimulus and consequence, makes for a binary world and cannot model the notion of "*strength of stimulus*" much discussed in models of behaviour in psychology. Each module merely does its thing as best as it can [14]. However with binary stimuli there is a limit to how well a module can do. For example, Skinner's law of threshold states that to elicit a response, the intensity of the stimulus must exceed some threshold. According to Skinner's law of magnitude, the magnitude of the response is a function of the intensity of the stimulus. Both these notions cannot be accommodated in predicate-based models.



The potential field based behaviour model of Arkin [3] [4] [5] and Anderson & Donath [1], is the closest to the "continuum stimulus" model of the psychologist. Here it is easy to model the *strength of stimulus*; the stronger the potential field gradient, the faster you run; if less fuel is left, use shorter paths, going closer to obstacle boundaries. One of the properties of the potential fields is that of superposition, which provides a convenient mechanism for combining the output from multiple behaviours which may operate concurrently. However, in its current form this model does not have the locality property. Some of the behaviour models are not purely reactive since the internal state of robot (fuel, temperature etc.) is taken into account in adaptation. While this is an important step in making the robot more adaptive, the potential fields model suffers from local minima and cyclic behaviour. Also, potential fields are limited to spatial tasks and cannot be adapted to reasoning at more abstract levels. However removal of cycles in continuum models causes a lesser decrease in power, usefulness and flexibility as compared to predicate models.

6.2 Fuzzy Logic

There is a dichotomy between potential field models, which are used almost exclusively in spatial path planning, and the predicate logic models which are used in reasoning, and do not ordinarily admit of grey levels. One approach for making the predicate models more flexible would be to use fuzzy logic; fuzzy stimuli to some extent realize the notion of strength of stimulus. For example, consider any predicate, say $\text{graspable}(x)$. There are always situations where its truth or falsehood are not very clear. Usually boolean functions use an arbitrarily defined threshold to define it. The $\text{graspable}(x)$ predicate has many nuances - where is the object CG when you grasp it? Is the moment just barely smaller than the maximum exertable moment, or is it much less? This would be partitioned into some fuzzy categories such as *almost-graspable*, *barely-graspable* etc.



The fuzzy categories then result in a strength of stimulus measure.

Fuzzy logic has been used to define fuzzy control functions, where both the stimulus and the consequence are expressed as fuzzy sets [6] [30] [44]. If-then rules are established using these fuzzy concepts. The robot has a knowledge base consisting of these rules, which may be thought of as defining a behaviour module. One such robot [44] performed quite well in the AAAI-93 robot competition, in the office rearrangement task which involved obstacle avoidance and goal seeking behaviour. However, changing the representation to fuzzy models will not eliminate cycles, and indeed the search space for detecting cycles may be larger.

6.3 Connectionist Architectures

Behaviour modules can also be modeled as connectionist networks of input-output nodes. These are structurally close to the biological and psychological models. Often a combination of specific stimuli are found to evoke a certain response. What is important is not that all the stimuli are present, but that some weighted sum of the stimuli is greater than some threshold value.

Maes proposes an action selection theory which allows arbitration among goals and actions based on inhibitory and excitatory connections between the actions that a module can perform [37]. Cyclic conflicts are reported in this architecture too. There is no description of the past "search", i.e. no "memory" of past states neither locally, nor globally, as a consequence the same planning mistake can be made over and over again. Maes [37] proposes a very simple solution of introducing some randomness in the system or using a second network to monitor possible loops or deadlocks in the first network, which is similar to a meta-level reasoner. But this randomness may be a temporary solution. A connectionist architecture to simultaneously satisfy the constraints of multiple independent behaviours has been proposed where a winner-take-all network can select the



most activated unit, for example, deciding turn commands to the vehicle [46].

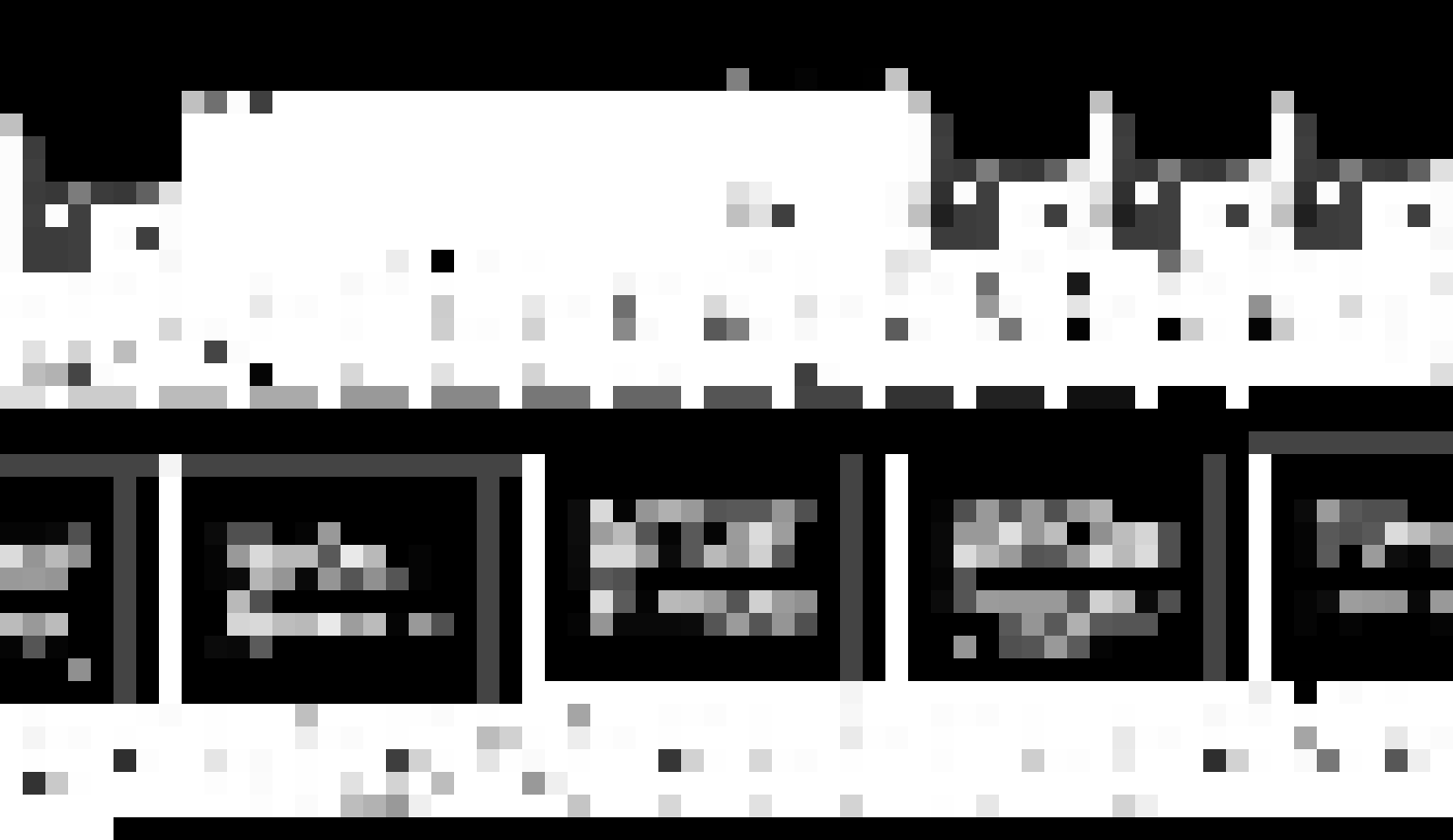
6.4 Hybrid Systems

Hybrid systems offer a compromise by employing a reactive system for low-level control and a planner for higher level decision making. Payton [45] builds a hierarchy of control in which lower level modules perform tasks requiring greatest immediacy while higher-level modules perform tasks involving greater assimilation of sensor data. Cycles can be avoided in these hybrid systems because of use of meta-knowledge. Instead of separating the reactive and traditional planning parts of the control system, a fully integrated reactive system is presented by [40] which allows internal representations.

6.5 Meta Rules

Meta rules are used in expert systems for resolving conflicts between rules that can be fired. One example is refraction: once a rule has fired, it may not fire again until the elements that match its conditions have been modified, or specificity: a more specific rule is preferred to a general rule [35]. Some more strategies are discussed [16]. In the domain of behaviour-based robotics the use of meta-level reasoners or hierarchical planners to avoid conflicts has been proposed, failure handlers and local planning modules selecting sets of behaviours that are appropriate for different situations and goals are used [45], but this violates the modularity assumption. As Kirsh has argued, tasks requiring knowledge that can be obtained by reasoning rather than perception cannot be done in behaviour-based robotics without representation [32].

How would meta-rules be used to avoid cyclic conflicts? For example, one could define a meta-rule that looks for repetitive rule applications for detecting cycles. More specific meta-rules, e.g.

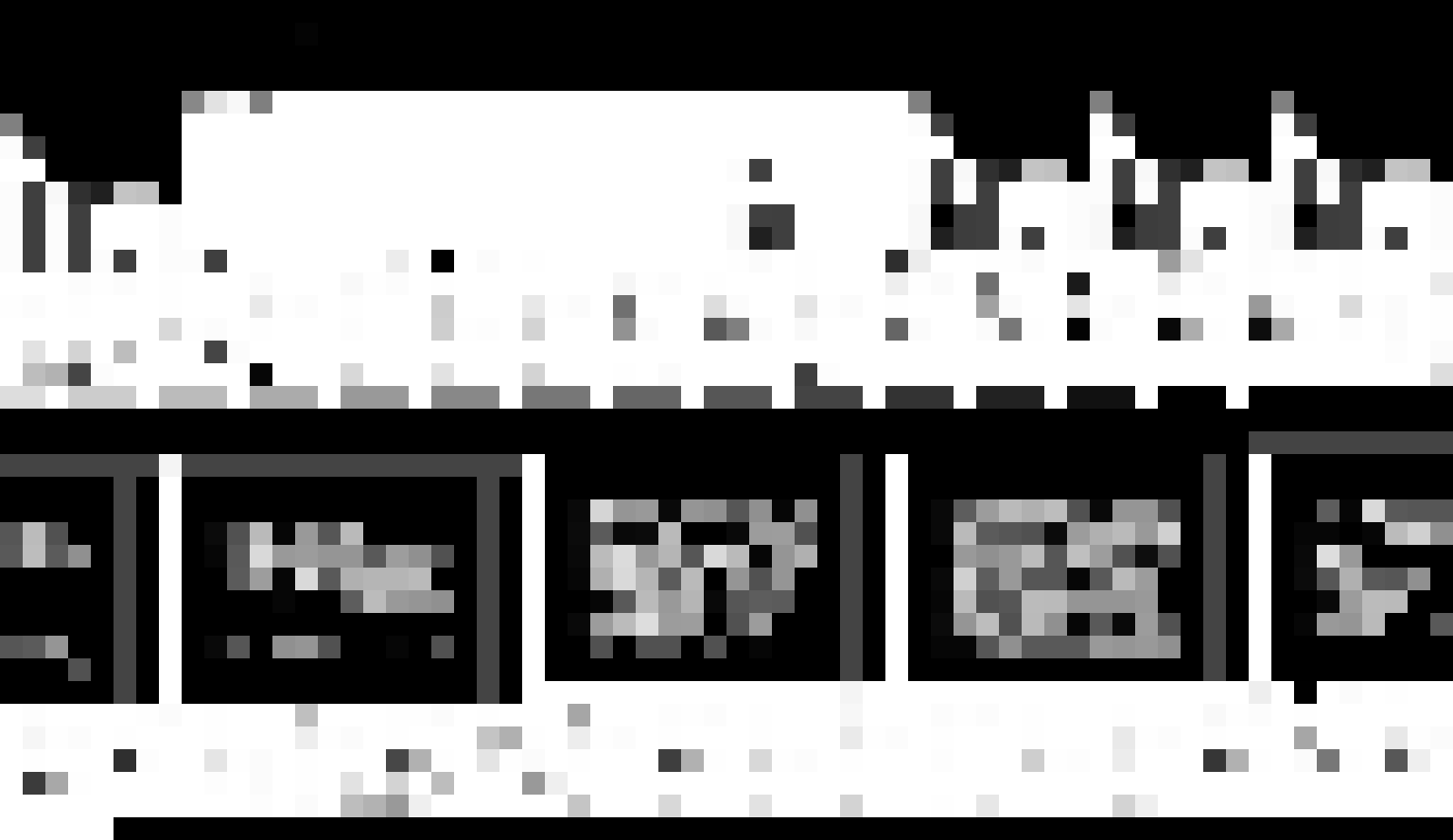


in the soda can example, would contain the information that the goal has to be achieved only once for a given soda can. However this would require us to track each can leading to inefficiency. However meta-rules are more difficult to design for reactive systems since the stimulus cannot be known beforehand, and all behaviours are always active. Another problem is that meta-rules can be quite complex, and as in knowledge-based systems, may result in conflicts of their own.

6.6 Learning and Behaviour Modification

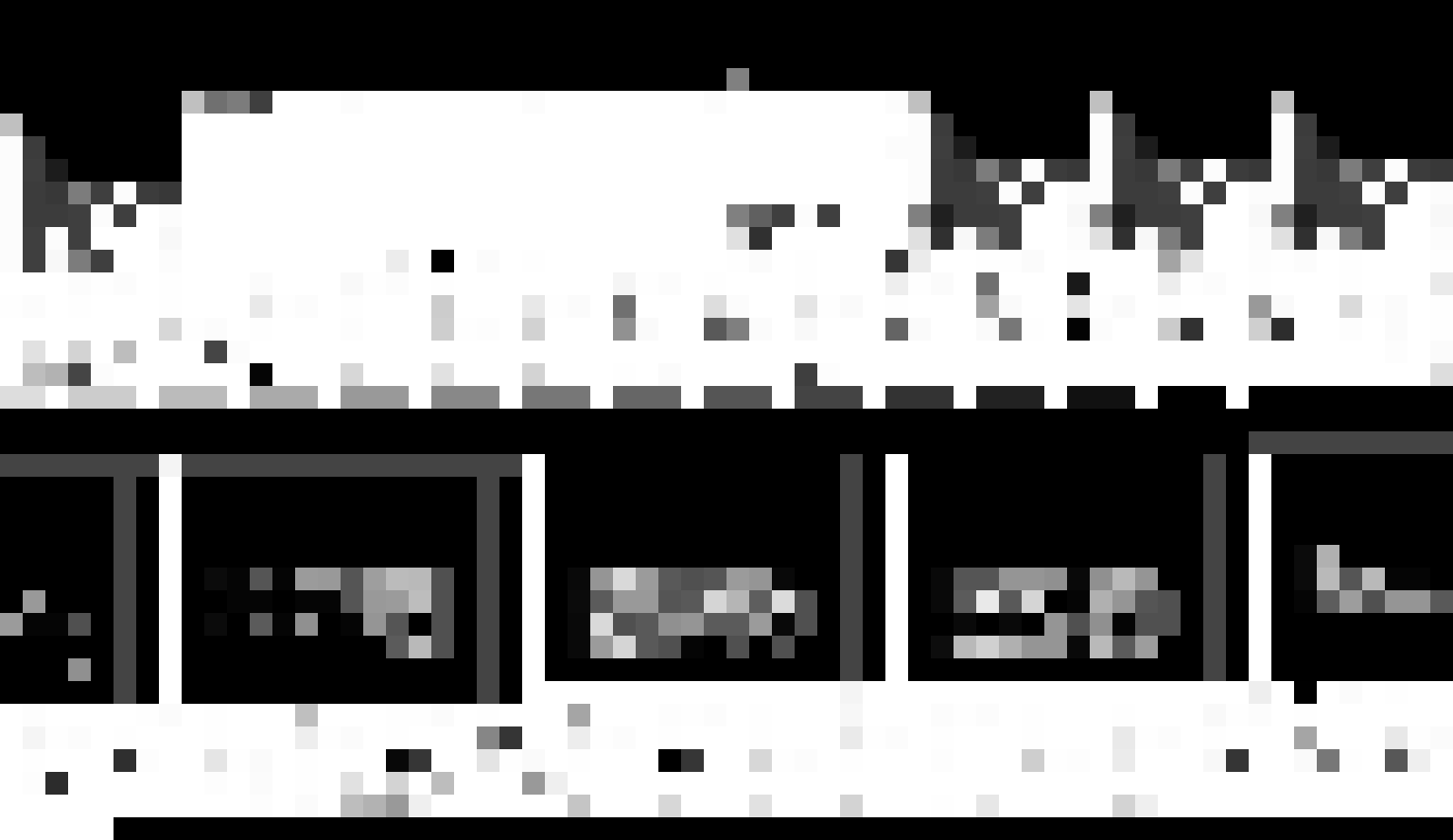
An important difference between robot behaviours and psychological models is the capacity to learn. One model for motor learning in humans is the schema model; here initial conditions, response specifications, sensory data and response outcome are stored after each movement. The strength of relationship among them and the accuracy increases with each successive movement. The learning stage uses feedback from the environment and is therefore a closed-loop. On the other hand, the models of [9] and [2] are open loop in which improvements can only be programmed by the behaviour designer.

In biological models of behaviour also, there are some behaviours that are genetically programmed (innate), and some that are learned (conditioned). Innate behaviours give an animal, an initial repertory of survival tools, but behaviour conditioning allows it to adapt to flexible environmental conditions. Current robotic models of behaviour represent only the innate aspects. Recent models of behaviour are beginning a discussion on learning; e.g. claiming that adaptation takes place only in the mind of the designer in the current approaches to behaviour-based robotics, ideas from ethology, machine learning and behaviour-based robotics are combined to propose a genetics-based learning architecture for an autonomous agent [21]. An algorithm that allows a behaviour-based robot to learn when to activate its behaviours (only conjunctive structures) on



the basis of positive and negative feedback has been proposed [38]. Others use a three level hierarchy (knowledge level, perceptuo-motor level and sensori-actuator level) to filter out abstractions [29].

Often the relative importance of goals will be context dependent. However pure perception cannot identify context, and this is poorly handled in purely reactive architectures. On one occasion, movement of the hand may be one without intention whereas on an other occasion it may be indicative of a goodbye [32]. This brings us to the role of *will*, of which *intention* is a weak version. Animals have the capacity of evaluating consequences of their actions, and those with pleasing consequences are repeated, resulting in conditioning. The question of will raises deep philosophical issues, the threshold of which AI is yet to cross. However, defining self-modifying reactive systems will certainly call for some model of intention or motivation, which will constitute part of the internal state of the robot. How this will affect behaviours, and how it can affect the incidence of cycles as a meta-level mediator is one of the major research directions for behaviour systems.



Chapter 7. Conclusion

The model presented in this paper reflects the most serious type of conflict in behaviour systems in the form of an infinite temporal cycle in the temporal execution of behaviour modules. A simple test is provided to detect such cycles, so that designers will not have nasty surprises awaiting them after implementation. We show that approaches such as prioritization will not avoid cycles.

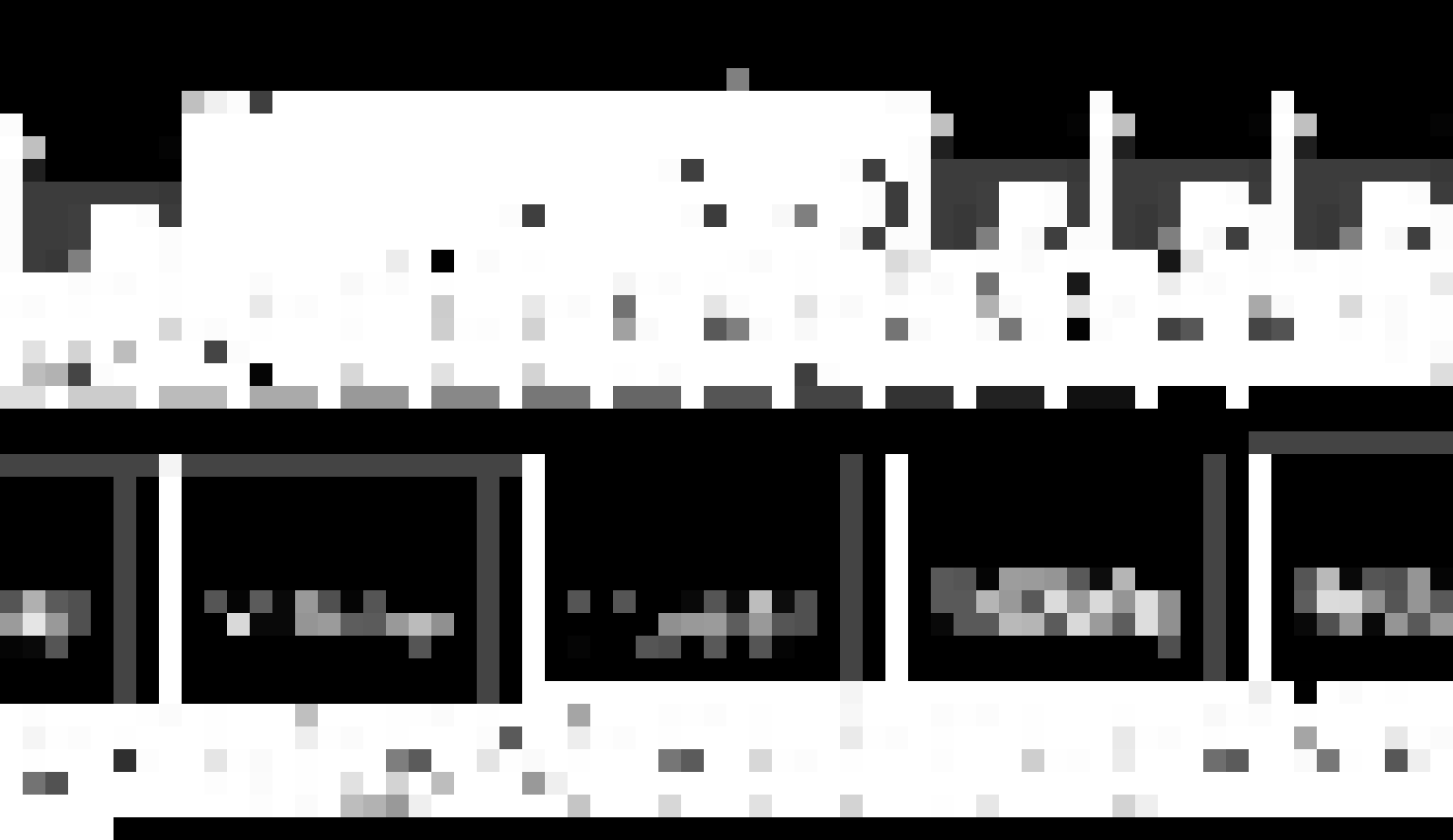
The key contribution of this work is to highlight an important limitation of the behaviour-based robotics approach. Cycles in the behaviour chains challenge the assumption regarding the modularity of behaviours which is fundamental to the basic advantages of behaviour-based models.

Strategies have been identified to detect and eliminate such cyclic conflicts in robot behaviour sequences. These involve behaviour redesign, and can use either response generalization, or stimulus specialization. This leads to an important conclusion regarding behaviour-based models, since the less general the stimulus, the less useful is the behaviour in the general context. Similarly, the more general the consequence of an action, the less constraining the effect i.e. less has been achieved by the behaviour. Both these changes add costs and inflexibility to the behaviour structure.

The principal insight to be gained from this discussion is that in behaviour design, there is a tradeoff between the power of a behaviour and the likelihood of cycles. The crucial task of the behaviour designer is to achieve just the right amount of refinement, without involving conflicts and without sacrificing too much flexibility. To what extent is this possible? Is there a limit on the complexity of tasks that can be performed without any central representation? How can meta-level mediators avoid conflicts? What is the role of learning and internal state? Some researchers (e.g. [23]), feel that internal state can be used to avoid endless futile loops. We feel that using internal state would not, in itself be able to remove this type of conflict, although it would make it easier to



modify the behaviours so that the conflict can be avoided. Another issue related to internal state is the intention of the robot (psychologists read *will*). Knowing the intention at some meta-level, it may be possible to construct tests for detecting conflicts, and even possibly of avoiding them. At the same time, models involving will or intention (as in Searle) are one of the most debated and difficult quagmires in AI today. A deeper question raised by the presence of such cycles in behaviour-based robotics, as well as in other branches of AI, is that of its significance to the entire search for artificial intelligence. Is there some bound on the complexity of any system claiming intelligence, before it begins to develop cyclic conflicts? These are but some of the questions raised by this research and the answers are sure to affect the future of the behaviour-based robot modeling paradigm in particular and that of representationless models of intelligence in general.



References

- [1] Tracy L. Anderson, Max Donath, Animal behaviour as a paradigm for developing robot autonomy, *Robotics and Autonomous Systems* 6(1 & 2) (1990) 145-168.
- [2] Ronald C. Arkin, Motor schema based navigation for a mobile robot: An approach to programming by behaviour, in: *Proceedings IEEE international conference on Robotics and Automation*, Raleigh, NC(1987) 264-271.
- [3] R.C. Arkin, E.Riseman and A.Hanson, "AuRA: An architecture for vision-based robot navigation", in: *Proceedings DARPA Image Understanding Workshop*, Los Angeles, CA(1987) 417-431.
- [4] Ronald C. Arkin, Behaviour-based robot navigation for extended domains, *Adaptive Behaviour* 1(2) (1992) 201-225.
- [5] Ronald C. Arkin, Dynamic replanning for a mobile robot based on internal sensing, in: *Proceedings IEEE International Conference on Robotics and Automation*, Arizona(1989), 1416-1421.
- [6] Anupam Bagchi, Himanshu Hatwal, Fuzzy logic-based techniques for motion planning of a robot manipulator amongst unknown moving obstacles, *Robotica*, 10(1992) 563-573.
- [7] Randall D. Beer, Hillel J. Chiel and Leon S. Sterling, A biological perspective on autonomous agent design, *Robotics and Autonomous Systems*, 6(1 & 2) (1990) 169-186.
- [8] D.E. Blackman, Images of man in contemporary behaviourism, in: Antony J. Chapman and Dylan M. Jones, eds., *Models of Man*, (The British Psychological Society, 1980) 99-112.
- [9] R.A. Brooks, A robust layered control system for a mobile robot, *IEEE J. Rob. Autom.* 2(1) (1986) 14-23.
- [10] Rodney A. Brooks, Intelligence without representation, *Artif. Intell.* 47(1-3) (1991) 139-159.
- [11] Rodney A. Brooks, A robot that Walks; Emergent behaviours from a carefully evolved network, *Neural Comput.* 1(2) (1989) 253-262.
- [12] Rodney A. Brooks, Elephants don't play chess, *Robotics and Autonomous Systems*, 6(1 & 2) (1990) 3-15.
- [13] Rodney A. Brooks, The whole iguana, in: Michael Brady, ed., *Robotics Science*, System Development Foundation Benchmark Series, (MIT Press, 1989) 432-456.
- [14] Rodney A. Brooks, A layered intelligent control system for a mobile robot, in: O.D.Faugeras and Georges Giralt, eds., *Robotics Research*, The Third International Symposium, (MIT Press, 1986) 365-372.
- [15] Brooks R.A., Connell J.H., Asynchronous distributed control system for a mobile robot, in: *Proceedings SPIE*, Cambridge, MA (1986) 77-84.
- [16] B.G. Buchanan, E.H. Shortliffe, *Rule-Based Expert Systems*, (Addison-Wesley Publishing Company, Inc., 1984).



- [17] Raja G. Chatila, Review of [9], in: Oussama Khatib, John J. Craig and Thomas Lozano-Perez, eds., *The Robotics Review 1*, (MIT Press, 1989) 103-108.
- [18] J.H. Connell, *Minimalist Mobile Robotics, A Colony Style Architecture for an Artificial Creature*, (Academic press Inc., Harcourt Brace Jovanovich Publishers, 1990).
- [19] Peter W. Cudhea, R.A. Brooks, Co-ordinating multiple goals for a mobile robot, in: *Preprints of Intelligent Autonomous Systems*, (North-Holland, Amsterdam, 1986), 168-174.
- [20] Narsingh Deo, *Graph Theory with applications to engineering and computer science*, (Prentice Hall of India Private Limited, New Delhi, 1989).
- [21] Marco Dorigo, Uwe Schnepf, Genetics-based machine learning and behaviour-based robotics: A new synthesis, *IEEE Trans. Syst. Man Cybern.* 23(1) (1993) 141-154.
- [22] Richard E. Fikes, Nils J. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, *Artif. Intell.* 2 (1971) 189-208.
- [23] Eran Gat, Robust low-computation sensor-driven control for task-directed navigation, in: *Proceedings IEEE International Conference on Robotics and Automation*, Sacramento, CA(1991) 2484-2489.
- [24] Eran Gat, On the role of stored internal state in the control of autonomous mobile robots, *AI Mag.* 14(1) (1993) 64-73.
- [25] Michael P. Georgeff, Planning, *Annual Review of Computer Science*, 2 (1987) 359-400.
- [26] Gould J.L., *Ethology: The Mechanisms and Evolution of Behaviour*, (W.W. Norton & Company, New York, 1982).
- [27] Ralph Hartley, Frank Pipitone, Experiments with the subsumption architecture, in: *Proceedings IEEE International Conference on Robotics and Automation*, Sacramento, CA(1991) 1652-1658.
- [28] Patrick J. Hayes, The frame problem and related problems in artificial intelligence, in: James Allen, James Hendler and Austin Tate, eds., *Readings in planning*, (Morgan Kaufmann Publishers, Inc., 1990) 588-595.
- [29] Henry Hexmoor, Johan Lammens, Guido Caicedo, and Stuart C. Shapiro, Behaviour based AI, cognitive processes, and emergent behaviours in autonomous agents, *A slightly revised version of a paper to appear in the proceedings of AI In Engineering*, Toulouse, France (1993).
- [30] Kaoru Hirota, Yoshinori Arai, Witold Pedrycz, Robot control based on membership and vagueness, in: M.M. Gupta, A. Kandel, Wyllis Bandler and Jerzy B. Kiszka, eds., *Approximate Reasoning In Expert Systems*, (Elsevier Science Publishers B.V., 1985) 621-635.
- [31] Lee Hudson and the editors of Time-Life books, *How We Learn?* (Time-Life books, New York, 1975).
- [32] David Kirsh, Today the earwig, tomorrow man?, *Artif. Intell.* 47(1-3) (1991) 161-184.



- [33] C. Ronald Kube, Hong Zhang, Collective robotics: From social insects to robots, *Adaptive Behaviour*, 2(2) (1994) 189-218.
- [34] Lempel A, Minimum feedback arc and vertex sets of a directed graph, *IEEE Transactions Circuit Theory*, 13(4) (1966) 399-403.
- [35] George F. Luger, William A. Stubblefield, *Artificial Intelligence and the Design of Expert Systems*, (The Benjamin/Cummings Publishing Company, Inc., 1989).
- [36] Peter Madison, Complex behaviour in natural settings, in: Mischel, ed., *Human Action, Conceptual and Empirical Issues*, (Academic Press, 1969) 223-259.
- [37] Pattie Maes, How to do the right thing? *Connection Sci.* 1(3) (1989) 291-323.
- [38] Pattie Maes & Rodney A. Brooks, Learning to co-ordinate behaviours, in: *Proceedings National Conference on Artificial Intelligence*, (1990) 796-802.
- [39] Pattie Maes, Situated agents can have goals, *Robotics and Autonomous Systems*, 6(1 & 2), (1990) 49-70.
- [40] Maja J. Mataric, Integration of representation into goal-driven, behaviour-based robots, *IEEE J. Rob. Autom.* 8(3) (1992) 304-312.
- [41] D. McFarland, *The Oxford Companion To Animal Behaviour*, (Oxford University Press, 1987).
- [42] David P. Miller, A twelve-step program to more efficient robotics, *AI Mag.* 14(1) (1993) 60-63.
- [43] Marvin L. Minsky, *The Society of Mind*, (Simon and Schuster, New York, 1986).
- [44] Illah Nourbakhsh et al, The winning robots from the 1993 robot competition, *AI Mag.* 14(4) (1993) 51-62.
- [45] David W. Payton, An architecture for reflexive autonomous vehicle control, in: *Proceedings IEEE Robotics and Automation Conference*, San Francisco, CA(1986) 1838-1845.
- [46] David W. Payton, J. Kenneth Rosenblatt and David M. Keirsey, Plan guided reaction, *IEEE Trans. Syst. Man Cybern.* 20(6) (1990) 1370-1382.
- [47] Richard A. Schmidt, A schema theory of discrete motor skill learning, *Psychol. Rev.* 82(4) (1975) 225-260.
- [48] Herbert A. Simon, *The Sciences of the Artificial*, (MIT Press, Cambridge, 1969).
- [49] Gerald J. Sussman, The virtuous nature of bugs, in: James Allen, James Hendler, and Austin Tate, eds., *Readings in Planning*, (Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990) 111-117.
- [50] Brian Yamauchi, Randal Nelson, A behaviour-based architecture for robots using real-time vision, in: *Proceedings IEEE International Conference on Robotics and Automation*, Sacramento, CA(1991), 1822-1827.

